INSTITUTO TECNOLOGICO
"CORDILLERA"

CARRERA DE ANÁLISIS DE SISTEMAS

OPTIMIZACIÓN DEL PROCESO DE FACTURACIÓN Y GESTIÓN DE PEDIDOS MEDIANTE UNA APLICACIÓN INFORMÁTICA EN EL RESTAURANTE REAL MANABITA

Proyecto de investigación previo a la obtención del título de

Tecnólogo Analista de Sistemas.

Autor: Ordoñez Yaguana Hernán Guillermo

Tutor: Ing. Richard Mafla

Quito, Noviembre 2013

DECLARATORIA

Declaro que la investigación es absolutamente original, auténtica, 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.

_____

Hernán Guillermo Ordóñez Yaguana

CC 110573175-4

CESIÓN DE DERECHOS

Yo, Hernán Guillermo Ordóñez Yaguana alumno de la Escuela de Sistemas, libre y voluntariamente cedo los derechos de autor de mi investigación en favor del Instituto Tecnológico Superior "Cordillera".

_____

CC: 110573175-4

AGRADECIMIENTO

Agradezco a todo el personal docente, por su apoyo y a todas aquellas personas que de una u otra forma me ayudaron a crecer como persona y como profesional.

DEDICATORIA

A mi familia por su apoyo incondicional que me han brindado en los momentos difíciles de mi carrera y mi vida en general.

Índice general

Índice de tablas

Índice de figuras

Índice de Anexos

Resumen ejecutivo

En el presente trabajo de grado, se desarrolla una aplicación que funcione en dispositivos móviles Android, con la cual se optimiza el proceso de facturación y gestión de pedidos en el restaurante Real Manabita. El proyecto hace uso de las tecnologías de la información y la comunicación con la finalidad de proporcionar una estructura de datos que facilite el proceso de recaudación y consolidación de información para la toma de decisiones dentro de la empresa.

Para el almacenamiento de la información se utiliza el motor de base de datos SQL Server, el cual nos brindara la seguridad necesaria para la empresa. El software es desarrollado en Visual Studio 2010 para el servicio web y la aplicación en Android 4.0. Los reportes generados de la base de datos permitirán ver el rendimiento de la empresa, mostrando las ventas y gastos online, lo cual ayudará a mejorar la producción y atención al cliente.

Al ser una nueva tecnología dentro de la empresa, este proyecto tiene como finalidad dar a conocer este tipo de proyecto como una solución que mejore y optimice el proceso de facturación y toma de pedidos, incrementando el nivel de servicio al cliente, asegurando entregas a tiempo y forma, surgiendo la posibilidad de la expansión del proyecto.

Como conclusión del presente proyecto se interpreta y unifica todas las variables en cuestión, determinando la opción más adecuada, para el funcionamiento correcto de la empresa. A sí mismo la aplicación se desarrolla para que le usuario final pueda adaptarse y le proporcione mayor capacidad de respuesta, según las necesidades y requerimientos de los clientes finales.

Abstract

In this paper, degree , developing an application that runs on Android mobile devices with which streamlines the billing and order management in the restaurant Real Manabita. El project makes use of information technologies and communication in order to provide a data structure that facilitates the process of collection and consolidation of information for decision -making within the company.

For information storage engine uses SQL Server database, which will provide us with the necessary security for the company. The software is developed in Visual Studio 2010 for web services and application in Android 4.0. Reports generated from the database to see the performance of the company, showing sales and expenses online, which will help to improve the production and customer service.

Being a new technology within the company, this project aims to raise awareness of this type of project as a solution to improve and streamline the billing process and taking orders, increasing the level of customer service, ensuring timely deliveries manner, raising the possibility of expanding the project.

In conclusion of this project is interpreted and unifies all the variables in question, determining the most appropriate option for the correct operation of the company. The application itself is developed to that end user can adapt and provide more responsive, according to the needs and requirements of end customers.

Capítulo I: Antecedentes

1.01 Contexto

El restaurante Real Manabita se encuentra en la ciudad de Quito en el barrio de Calderón, funciona hace 5 años y se dedica a la venta de mariscos. Realiza la venta de toda clase de platos de comida, para todos los gustos y sabores, con el fin de satisfacer a todos sus clientes.

Actualmente en el restaurante, la toma de pedidos se realiza de forma manual, se llenan las facturas con letra ilegible o con errores, esto provoca que la factura sea anulada, se produzcan equivocaciones y retrasos en la entrega de pedidos a los clientes, a los cuales les causa muchas molestias. Los mismos tienen que facilitar sus datos cada vez que visitan el restaurante, causando pérdida de tiempo y retrasos en los pedidos.

En el restaurante el cliente, no conoce todos los platos de comida que se ofertan, lo que trae como consecuencia, que el mismo se tarda mucho tiempo para decidir lo que desea, retrasando al mesero en la realización su trabajo.

Al momento que el cliente desea su cuenta debe de realizarse manualmente, lo que provoca que se produzcan errores en los cálculos, que al momento de realizar un balance de ventas se produzca una información errónea de la producción de la empresa, causando decisiones equivocadas para mejorar los productos y las ventas de la empresa.

1.02 Justificación

La finalidad de la automatización de la gestión de pedidos, es ayudar a la organización y mejora de ventas del restaurante, empleando las tecnologías de la información y la comunicación.

En la actualidad vemos que existen muy pocos restaurantes que tienen automatizado, la gestión de pedidos y facturación con esto se logrará una excelencia en atención, optimización de recursos e incremento de la productividad.

Cuando se logre automatizar la toma de pedidos del restaurante, se tendrá un balance de ventas y productos en stock en tiempo real, debido a que todo ello será automático.

El software implantado ofrecerá una ejecución de pedidos en forma inmediata, los clientes ya no tendrán que esperar que el mesero les tome el pedido, lo podrán hacer ellos mismos desde su mesa con una Tablet, la cual se comunicara mediante Wi-Fi con la cocina para que los chefs puedan preparar los platos.

Se tendrá una facturación automática, con lo cual los clientes decidirán cuando imprimir su factura y pagar la cuenta, sin tener que hacer cola o esperar que se realice la misma.

Se espera obtener un balance de ventas en tiempo real, lo cual servirá para que los gerentes puedan tomar las decisiones correctas para la empresa.

La satisfacción del cliente aumentaría considerablemente ya que es una nueva forma de realizar sus pedidos en un restaurante, con lo cual aumentarán las ventas y los ingresos de la empresa.

Por todo lo indicado anteriormente, este trabajo es importante, ya que es una parte de la solución tecnológica, que mejorara la calidad del servicio, el aumento de las ventas y la satisfacción del cliente, justificando la presentación de este trabajo de investigación.

1.03 Matriz T

La siguiente matriz, muestra la construcción de un árbol de problemas donde se muestra un análisis de la situación actual, describiendo las fuerzas bloqueadoras y las fuerzas impulsoras que ayudan a mejorar la situación actual de la empresa.

Tabla 1.

Análisis de fuerzas T.

ANÁLISIS DE FUERZAS T					
Situación Empeorada	**Situación Actual**				**Situación Mejorada**
Pocas ventas y clientes insatisfechos	Retraso en pedidos y facturación manual				Mejoras en ventas y satisfacción de los clientes
Fuerzas impulsadoras	**I**	**PC**	**I**	**PC**	**Fuerzas bloqueadoras**
Toma de pedidos de forma ágil y rápida	2	5	3	5	Mala atención al cliente
Facturación de forma correcta e inmediata	3	5	3	5	Retraso en la elaboración de facturas y con errores
Clientes satisfechos	3	4	4	4	Usuarios descontentos
Balance de ventas y gastos que muestren la producción de la empresa	2	4	3	4	Desconocimiento de la producción de la empresa
Control de todos los productos que existen y que más se venden	2	4	2	4	Bajo control de productos más vendidos
Innovación de la empresa en venta de comida siguiendo procesos	2	4	3	4	Poca innovación de la empresa siguiendo procesos

En la matriz T, se puede observar que el retraso de pedidos y la facturación manual, nos pueden llevar a una situación mejorada o empeorada, se apoya en las fuerzas impulsoras que nos ayudan a mejorar la situación actual, usando las tecnologías de la información y la comunicación, mediante la optimización de los procesos existentes. Las fuerzas bloqueadoras obstaculizarían el proceso de cambio con lo cual la empresa tendría un bajo volumen de ventas y clientes descontentos.

Capítulo II: Análisis de Involucrados

2.01 Análisis de los involucrados

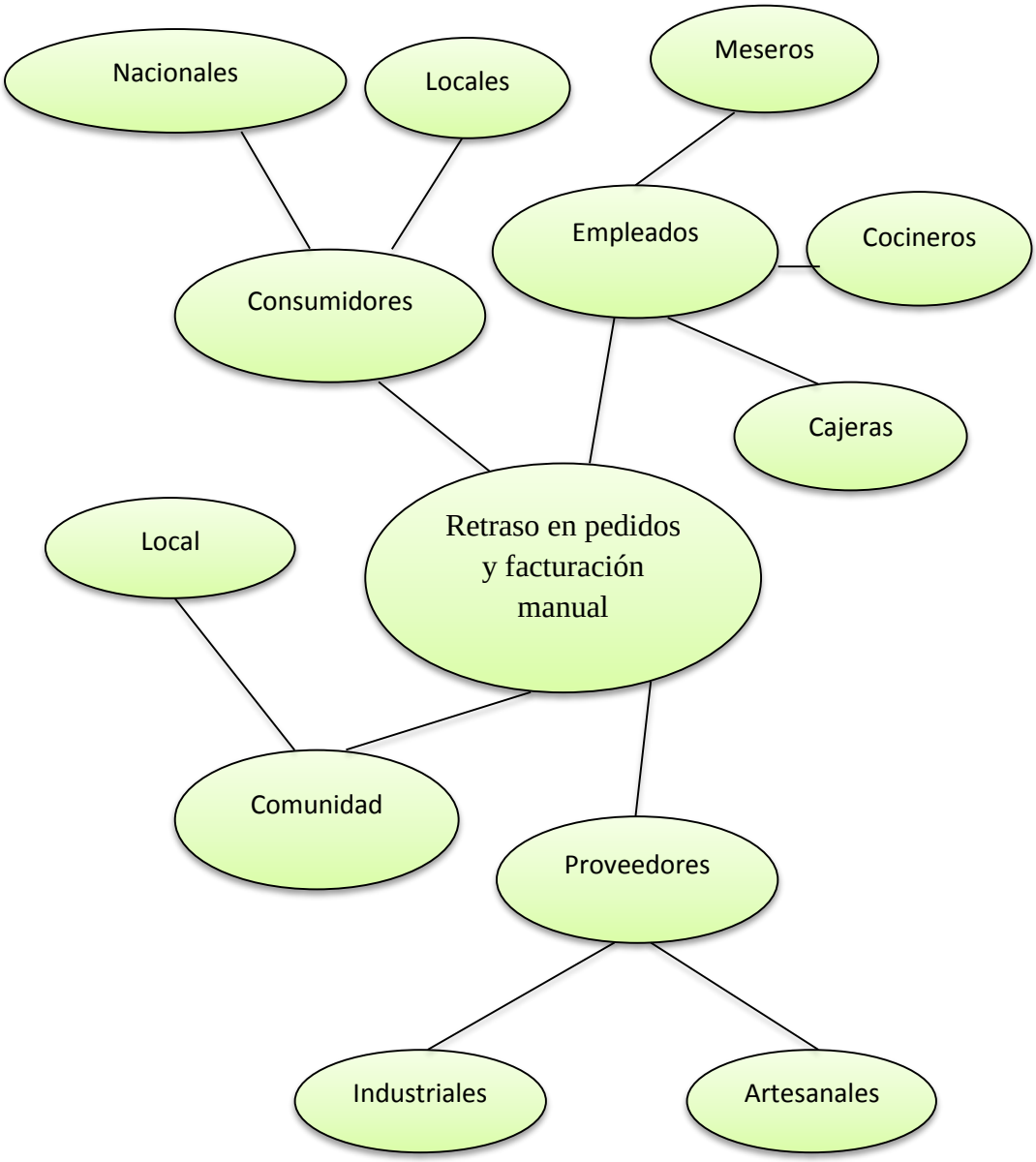


Figura 1. Muestra las personas o grupos que tienen intereses a favor o en contra del proyecto, y que a través de sus aptitudes o acciones, pueden influir para el éxito o fracaso del mismo.

2.02 Matriz de análisis de involucrados

Tabla 2.

Matriz de análisis de involucrados.

Actores Involucrados	Intereses sobre el problema central	Problemas Percibidos	Recursos Mandatos y Capacidades	Intereses sobre el Proyecto	Conflictos Potenciales
Consumidores	Ser bien atendidos	Retraso en pedidos y mala atención	Demanda actual y potencial	Sentirse a gusto e importantes	Otros negocios
Empleados	Optimizar el trabajo	Desorden en realización de pedidos y facturas	Formación en buena atención al cliente	Organización en el trabajo e incremento de salario	Empresas industrializadas
Proveedores	Mejorar sus ventas	Pedidos mal realizados	Abastecer productos constantemente	Oportunidades de incrementar su comercio	Nuevos mercados
Comunidad	Incrementar el turismo	Poca afluencia de personas	Promoción Turística	Creación de empleos	Aumento de impuestos

Podemos observar en las figuras, que la facturación y gestión de pedidos está directamente relacionado con los consumidores, empleados, proveedores y con la comunidad, los cuales están afectados directa o indirectamente por el proyecto. Además indica los mandatos, recursos, políticas y prioridades que son requeridos y deseados para poder dar la solución al problema.

Capítulo III: Problemas y Objetivos

En las siguientes figuras, se muestra un análisis de los problemas y los objetivos que se pretender resolver o alcanzar con el fin de llevar a cabo el proyecto.

3.01 Construcción del árbol de problemas

La siguiente figura describe el problema central, y analiza la relación entre causa y efecto del problema central y los problemas percibidos por los involucrados. Aquí podemos observar que el retraso en pedidos y facturación manual, tiene causas negativas las mismas que tienen efectos que no favorecen a la empresa para que mejoren sus ventas. (Ver anexo A.01).

3.02 Construcción del árbol de objetivos.

La figura siguiente describe los procesos que se quieren cumplir mediante un árbol de objetivos, convirtiendo el problema central en un propósito, sus causas en medios y sus efectos en fines. Se puede observar que el propósito es generar pedidos a tiempo y la facturación online, ayudado de las tecnologías de la información y la comunicación. (Ver anexo A.02)

Capítulo IV: Análisis de Alternativas

4.01 Matriz de análisis de alternativas

La matriz siguiente identifica las estrategias y alternativas a partir del árbol de objetivos, que si son ejecutadas, podrían promover el cambio de la situación actual a la situación deseada.

Tabla 3.

Matriz de análisis de alternativas.

Objetivos	Impacto sobre el Propósito	Factibilidad Técnica	Factibilidad Financiera	Factibilidad Social	Factibilidad Política	Total	Categorías
El stock es en línea	5	4	4	3	4	20	Alta
Los procesos de la empresa se llevan de forma automática	5	4	4	3	4	20	Alta
Los pedidos son exactos y online	5	5	4	3	4	21	Alta
Todos los cálculos son computarizados	5	5	4	3	4	21	Alta
La facturación es organizada, rápida y disponible a todo momento.	4	4	3	4	4	19	Alta
Los pedidos tienen una organización excelente utilizando tecnología	4	4	4	4	4	20	Alta
Total	28	26	23	20	24	121	

Se puede observar en la tabla que todos los objetivos tienen una categoría alta, con lo cual son factibles técnica, financiera, social y política, haciendo que el proyecto se pueda llevar a cabo.

4.02 Matriz de análisis del impacto de los objetivos

La tabla siguiente muestra un análisis de cuál es el impacto de la aplicación en los objetivos del proyecto, donde se obtiene la visión de la situación deseada y las estrategias que se aplicarán para conseguirla.

En la matriz se obtiene un puntaje alto, en factibilidad, impacto ambiental, relevancia y sostenibilidad, además se obtiene un alto grado de participación del género femenino, ayudando a igualdad de género. (Ver anexo A.03).

4.03 Diagrama de estrategias

A continuación se establece la estructura y alcance de las estrategias que intervienen en el proyecto, expresadas en conjuntos de objetivos que son considerados como factibles de realización, y están dentro de las posibilidades del proyecto.

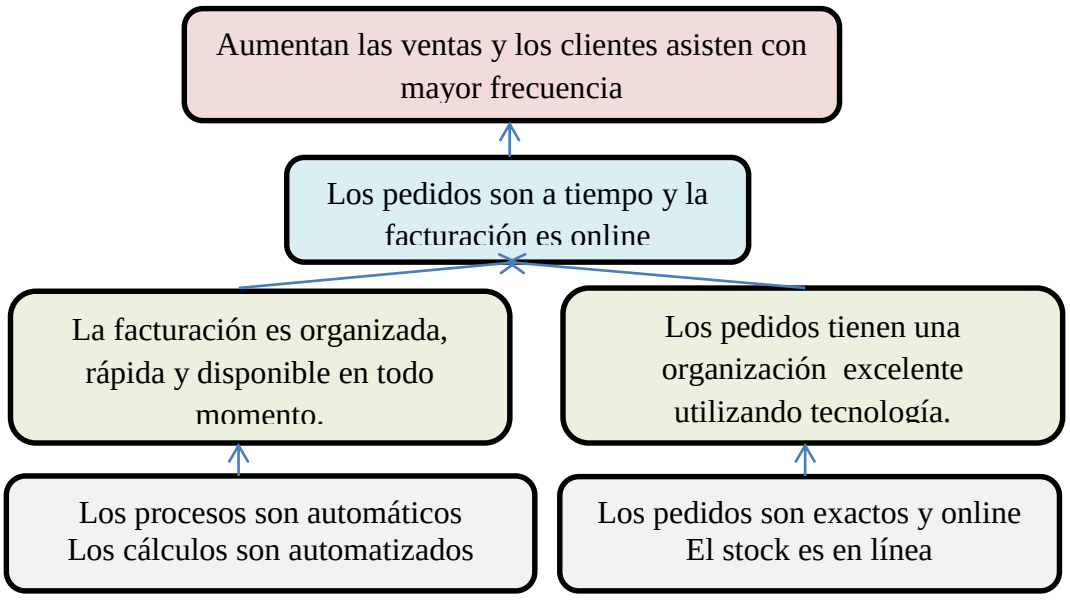


Figura 2. Analizando la figura observamos que la finalidad del proyecto es el aumento de ventas e incremento de clientes, que se logra mediante el propósito, que es los pedidos a tiempo y la facturación online, ayudado de los componentes y sus respectivos objetivos.

4.04 Matriz del marco lógico

En la siguiente matriz se describe que se desea lograr con el proyecto, como se alcanzarán el propósito y sus componentes, que factores externos son indispensables para el éxito del mismo, como se pretende medir el cumplimiento de los objetivos y resultados, como se puede obtener la información y finalmente que recursos son necesarios para la ejecución del proyecto.

La matriz describe el fin del proyecto: aumentan las ventas y los clientes asisten con mayor frecuencia, se especifica los indicadores, los medios de verificación y los supuestos necesarios para su realización.

Además se cita el propósito del proyecto, los componentes del proyecto y las actividades del proyecto, evaluando cada una de ellas para poder llevar a cabo la ejecución del proyecto. (Ver anexo A.04.01, A.04.02).

Capítulo V: Propuesta

5.01 Justificación del software

Para el funcionamiento del proyecto se utiliza un Servicio Web hecho en Visual Studio 2010 ASP.NET, C#, ya es una forma de integrar aplicaciones creadas en lenguajes y plataformas diferentes, a través de Internet o bien en una Intranet basándote en estándares para intercambiar datos entre aplicaciones.

Una segunda razón por la que los servicios Web son muy prácticos es que pueden aportar gran independencia entre la aplicación que usa el servicio Web y el propio servicio. De esta forma, los cambios a lo largo del tiempo en uno no deben afectar al otro. Esta flexibilidad será cada vez más importante, dado que la tendencia a construir grandes aplicaciones a partir de componentes distribuidos más pequeños es cada día más utilizada.

Como plataforma de Base de Datos tenemos SQL Server 2008 es una plataforma de base de datos para transacciones en linea (OLTP) a gran escala, bodegas de datos (data warehousing), y aplicaciones de comercio electrónico; a su vez es un plataforma de inteligencia de negocios con integración de datos, análisis, y soluciones de reporteo.

La plataforma de parte de los usuarios será Android 4.0 que ofrece una refinada interfaz unificada para teléfonos y tablets e introduce funciones innovadoras para los usuarios y desarrolladores.

Android 4.0 se basa en las cosas que a la gente le gusta de Android - fácil de realizar, múltiples tareas, notificaciones intuitivas, pantallas de inicio personalizables, widgets de tamaño variable, interactividad profunda - y añade potentes nuevas formas de comunicar y compartir.

Para realizar el consumo del Servicio Web desde Android se utiliza la librería ksoap2-android. En primer lugar hay que empezar diciendo que Android no incluye de forma nativa ningún tipo de soporte para el acceso a servicios web de tipo SOAP. Es por esto por lo que vamos a utilizar una librería externa para hacernos más fácil esta tarea. Entre la oferta actual, la opción más popular y más utilizada es la librería ksoap2-android. Esta librería es un fork, especialmente adaptado para Android, de la antigua librería kSOAP2. Este framework nos permitirá de forma relativamente fácil y cómoda utilizar servicios web que utilicen el estándar SOAP.

5.02 Análisis y diseño

5.02.01 Diagramas de caso de uso

El diagrama de casos de uso representa la forma en como un Cliente (Actor) opera con el sistema en desarrollo, además de la forma, tipo y orden en como los elementos interactúan (operaciones o casos de uso).

Un diagrama de casos de uso consta de los siguientes elementos:

Actor.

Una definición previa, es que un Actor es un rol que un usuario juega con respecto al sistema. Es importante destacar el uso de la palabra rol, pues con esto se especifica que un Actor no necesariamente representa a una persona en particular, sino más bien la labor que realiza frente al sistema.

Caso de Uso.

Es una operación/tarea específica que se realiza tras una orden de algún agente externo, sea desde una petición de un actor o bien desde la invocación desde otro caso de uso.

Relaciones.

Asociación.

Es el tipo de relación más básica que indica la invocación desde un actor o caso de uso a otra operación (caso de uso). Dicha relación se denota con una flecha simple.

Dependencia o Instanciación.

Es una forma muy particular de relación entre clases, en la cual una clase depende de otra, es decir, se instancia (se crea). Dicha relación se denota con una flecha punteada.

Generalización.

Este tipo de relación es uno de los más utilizados, cumple una doble función dependiendo de su estereotipo, que puede ser de Uso (<<uses>>) o de Herencia (<<extends>>).

Este tipo de relación está orientado exclusivamente para casos de uso (y no para actores).

El siguiente diagrama de caso de uso, *DCU-001*, expresa el funcionamiento de una forma global que actualmente se encuentra funcionado la empresa.

(Ver Anexo A.05).

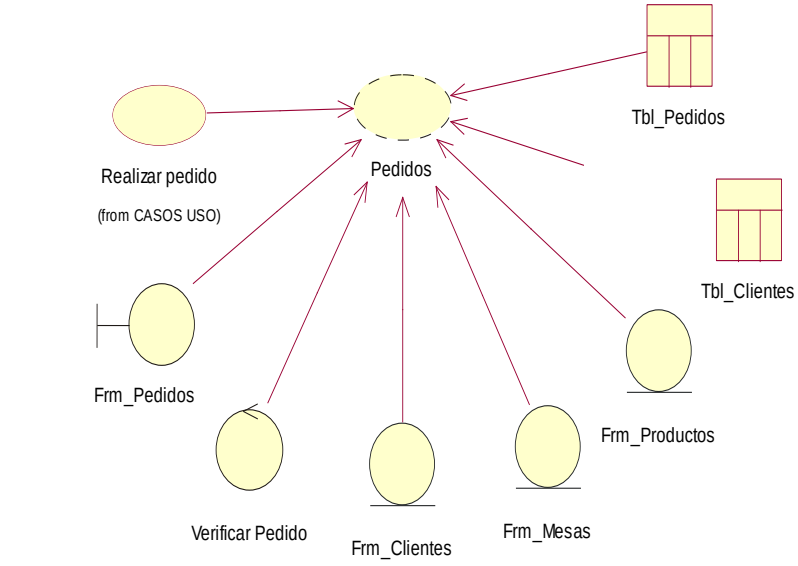


Figura 3. *DCR-001*. El siguiente diagrama de realización, describe el proceso de los pedidos.

Tabla 4.

Especificación de caso de uso realizar pedido.

Nombre	Realizar pedidos	Id: DCR-001
Actores	Mesero-Cliente	
Descripción	El caso de uso inicia cuando le cliente necesita hacer un pedido	
Flujo de eventos	1. El cliente realiza pedido 2. Mesero verifica pedido 3. Mesero registra pedido 4. Cliente verifica pedido	
Post condiciones	El cliente revisó los productos en stock	

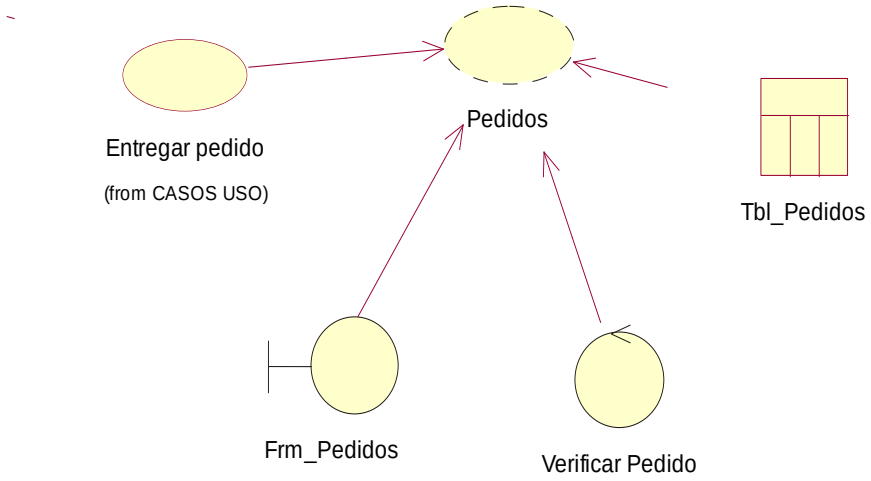


Figura 4. DCR-002. El siguiente diagrama de realización describe el proceso de entrega de pedidos

Tabla 5.

Especificación de caso de uso entregar pedido.

Nombre	Entregar pedidos	Id: DCR-002
Actores	Mesero-Cliente-Cocinero	
Descripción	El caso de uso inicia cuando el cocinero termina de preparar el pedido	
Flujo de eventos	1. Cocinero prepara pedido 2. El mesero entrega pedido 3. El cliente verifica pedido 4. Mesero registra pedido entregado	
Post condiciones	El cliente realizó correctamente el pedido	

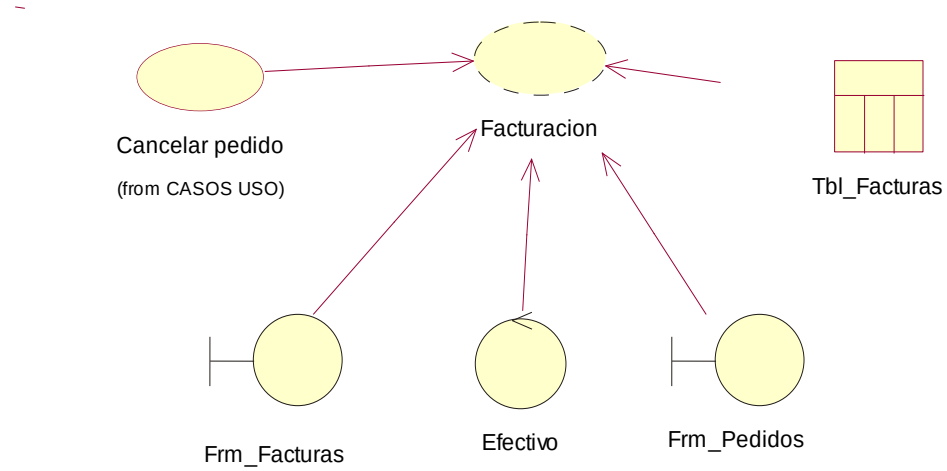


Figura 5. *DCR-003.* El siguiente diagrama de realización describe el proceso de cancelar los pedidos.

Tabla 6.

Especificación de caso de uso cancelar pedido.

Nombre	Cancelar pedidos	Id: DCR-003
Actores	Cajera-Cliente	
Descripción	El caso de uso inicia cuando el cliente desea pagar el pedido	
Flujo de eventos	1. El cliente solicita factura 2. Cajera realiza factura 3. Cliente paga el dinero 4. Cajera registra pago de pedido	
Post condiciones	El pedido debe de estar verificado	

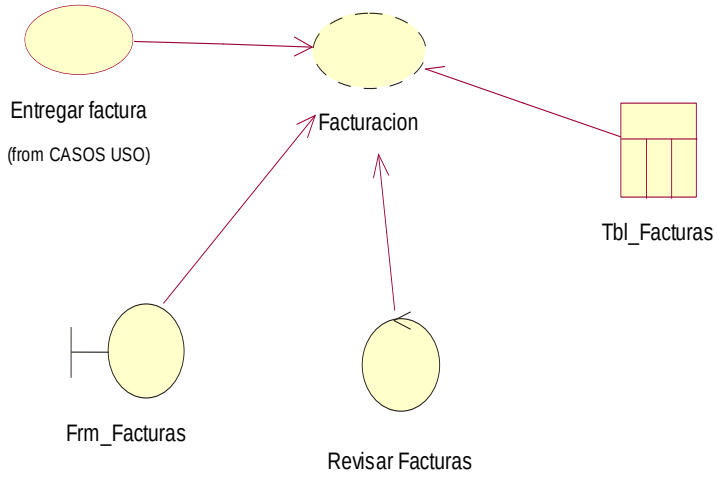


Figura 6. DCR-004. El siguiente diagrama de realización describe el proceso de entregar la factura al cliente.

Tabla 7.

Especificación de caso de uso entregar factura.

Nombre	Entregar factura	Id: DCR-004
Actores	Cajera-Cliente	
Descripción	El caso de uso inicia cuando el cliente solicita la factura	
Flujo de eventos	1. El cliente solicita factura 2. Cajera revisa factura 3. Cajera registra factura 4. Cliente recibe factura	
Post condiciones	Los datos del cliente están verificados	

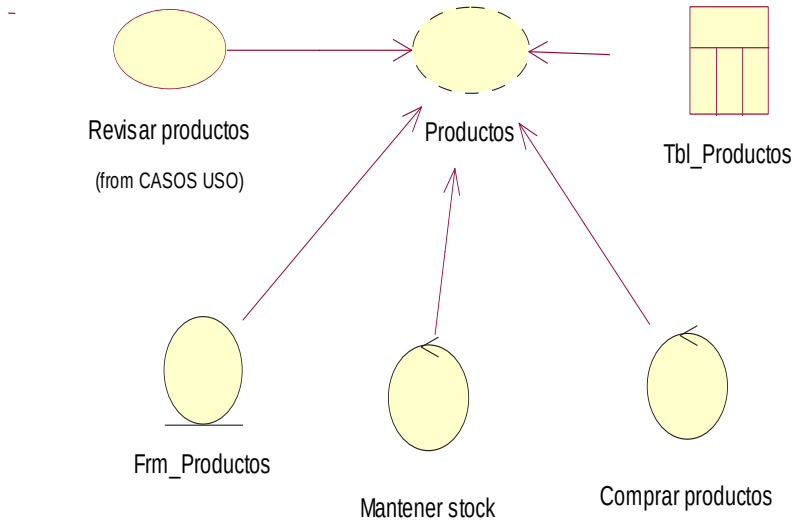


Figura 7. *DCR-005*. El siguiente diagrama de realización describe el proceso de revisar los productos para mantener el stock.

Tabla 8.

Especificación de caso de uso revisar productos.

Nombre	Revisar productos	Id: DCR-005
Actores	Cocinero-Bodeguero	
Descripción	El caso de uso inicia cuando el cocinero solicita productos	
Flujo de eventos	1. El cocinero solicita nuevos productos 2. Bodeguero verifica productos 3. Bodeguero compra productos 4. Bodeguero mantiene el stock	
Post condiciones	Los pedidos se realizan correctamente	

5.02.02 Diagramas de secuencia

Los diagramas UML de secuencia y de colaboración (llamados diagramas de interacción) se utilizan para modelar los aspectos dinámicos de un sistema. Un diagrama de interacción consiste en un conjunto de objetos y sus relaciones, incluyendo los mensajes que se pueden enviar entre ellos.

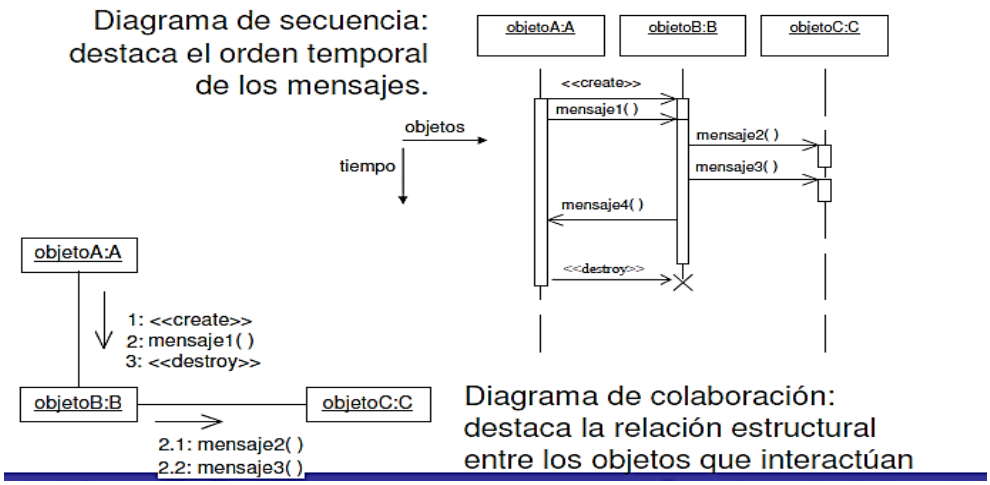


Figura 8. Los diagramas de secuencia destacan el orden temporal de los mensajes. Los diagramas de colaboración destacan la organización estructural de los objetos que envían y reciben mensajes.

Ambos diagramas (secuencia y colaboración) son semánticamente equivalentes. Se puede pasar de uno a otro sin pérdida de información.

En los diagramas de secuencia, la línea de vida de un objeto es la línea discontinua vertical, que representa la existencia de un objeto a lo largo de un periodo de tiempo. El foco de control es un rectángulo delgado que representa el periodo de tiempo durante el cual un objeto ejecuta una acción.

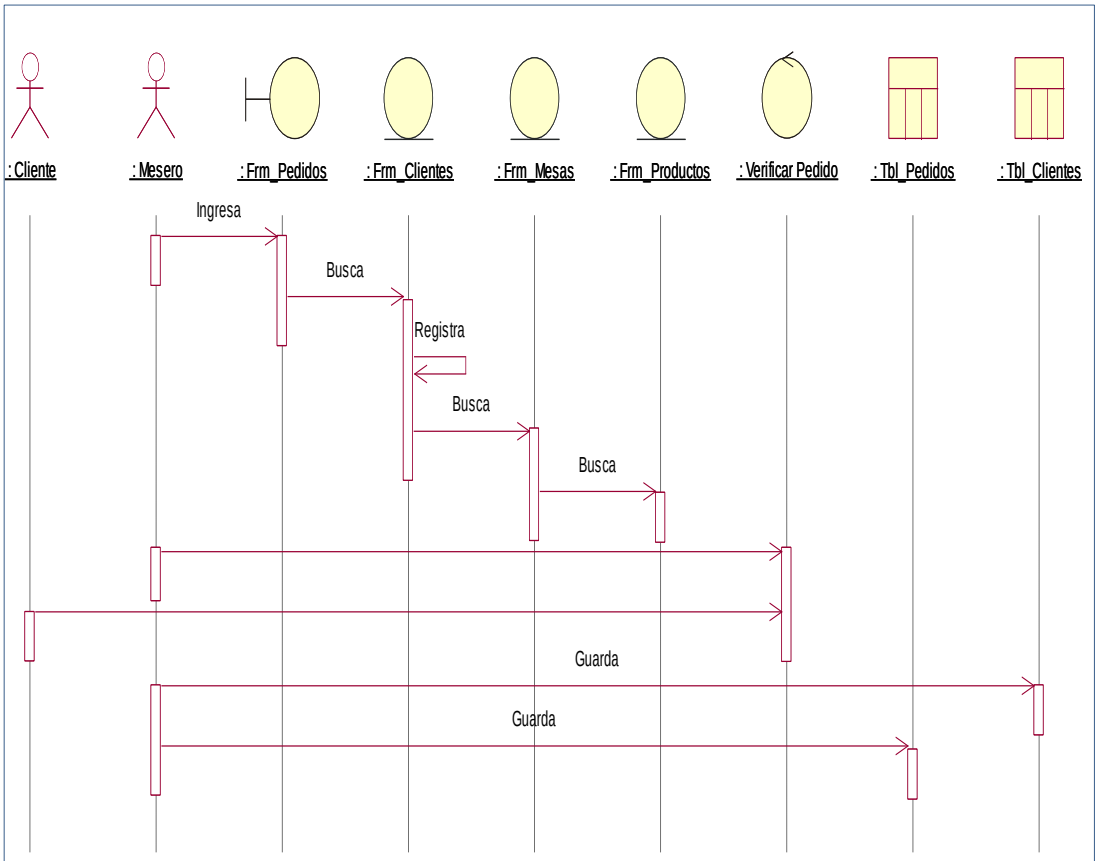


Figura 9. Diagrama de secuencia que muestra el proceso de realizarlos pedidos.

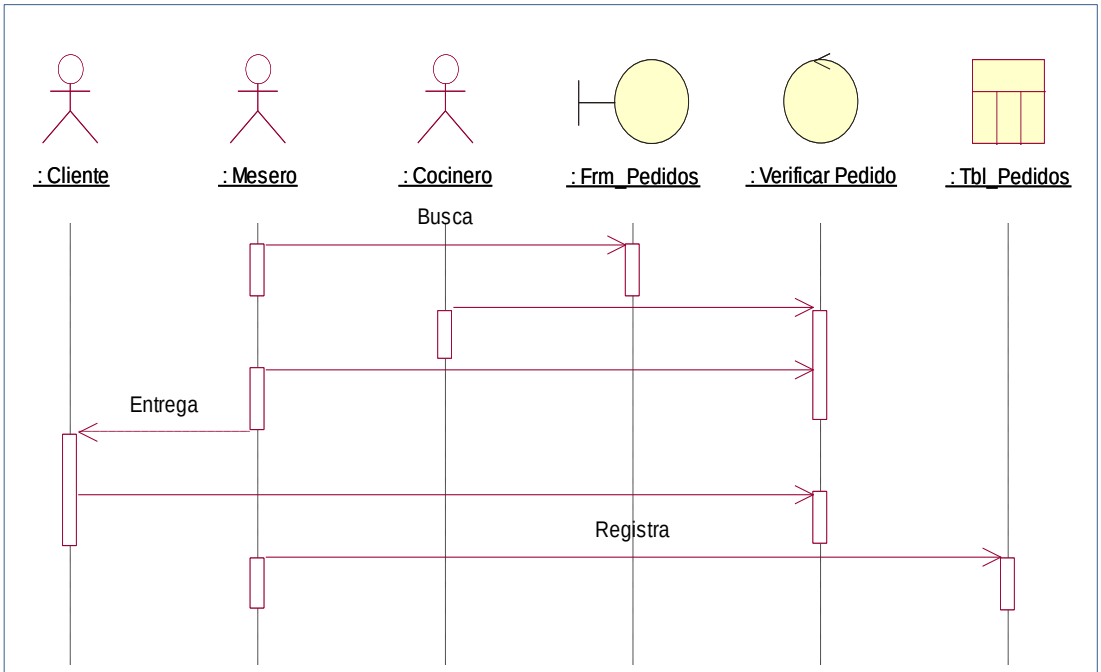


Figura 10. Diagrama de secuencia entregar pedido.

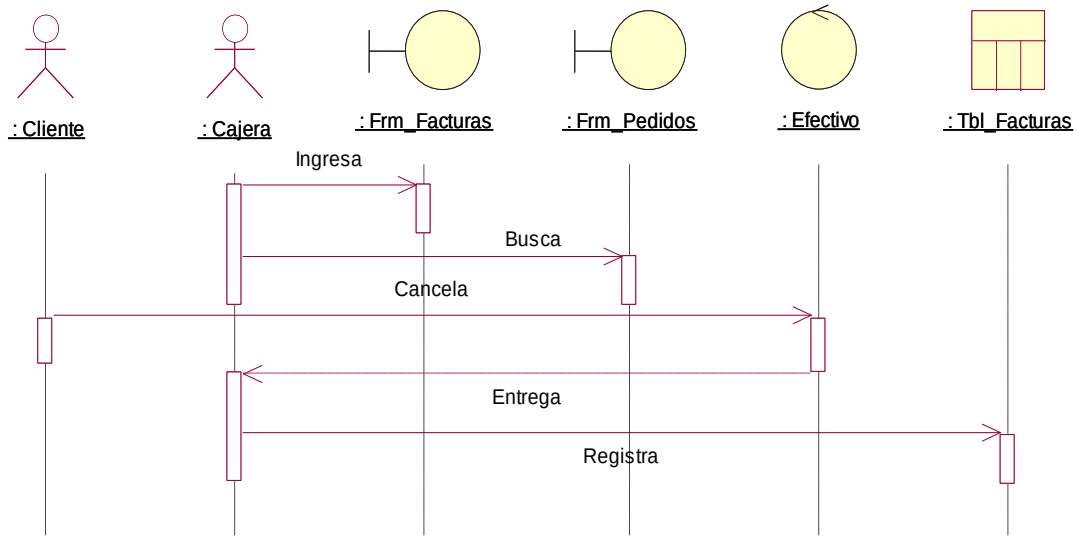


Figura 11. Diagrama de secuencia cancelar pedido.

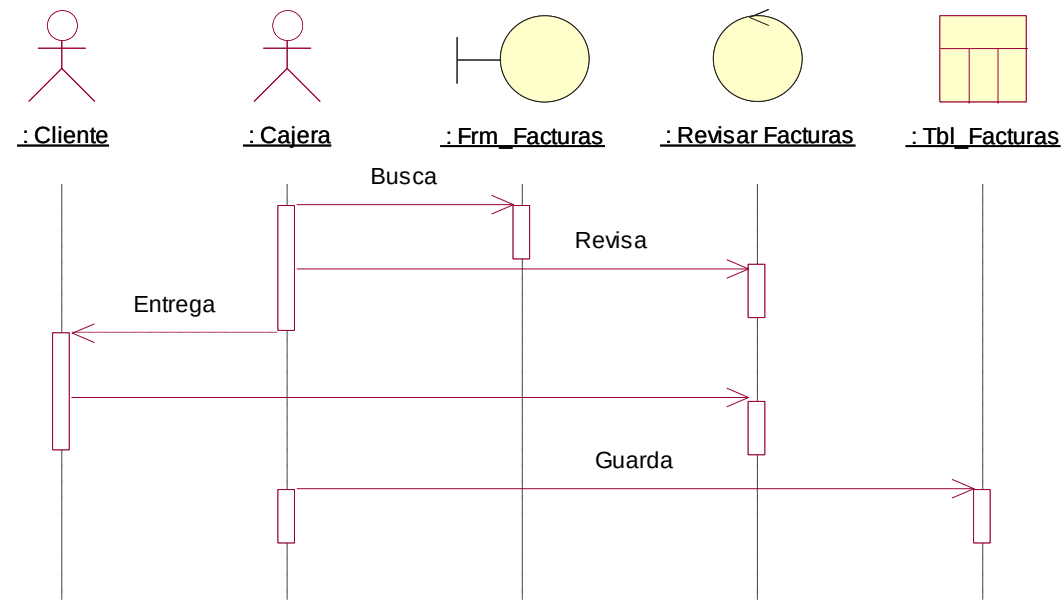


Figura 12. Diagrama de secuencia entregar factura.

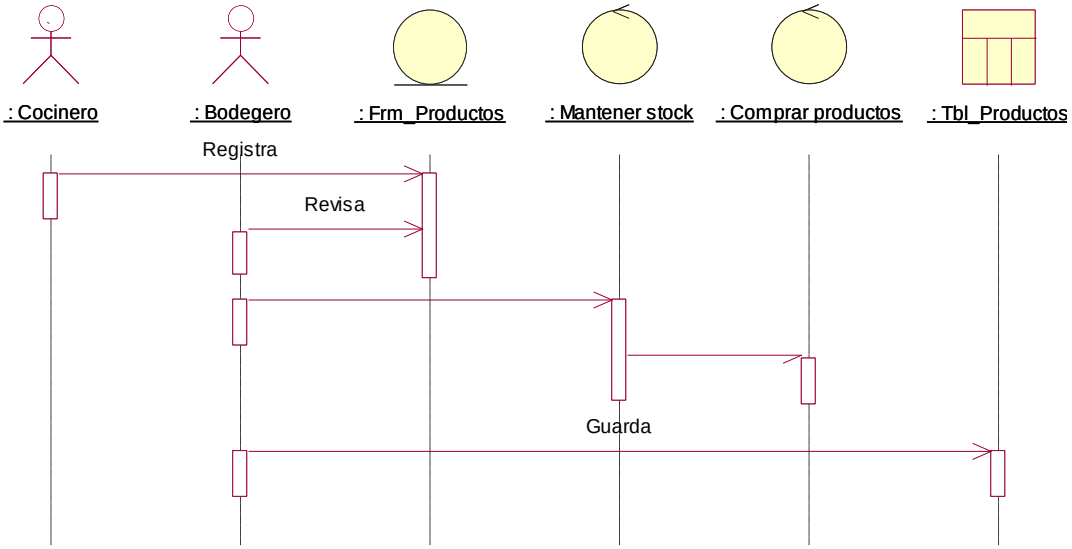


Figura 13. Diagrama se secuencia revisar productos.

5.02.03 Diagramas de colaboración

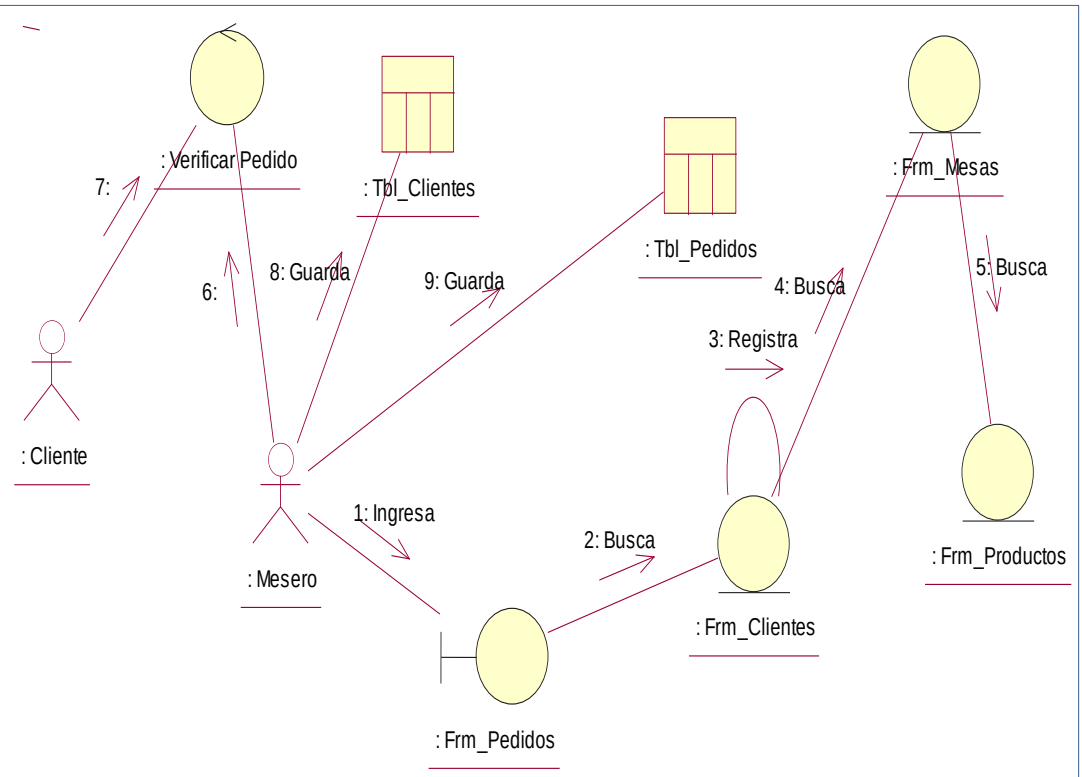


Figura 14. Diagrama de colaboración realizar pedido.

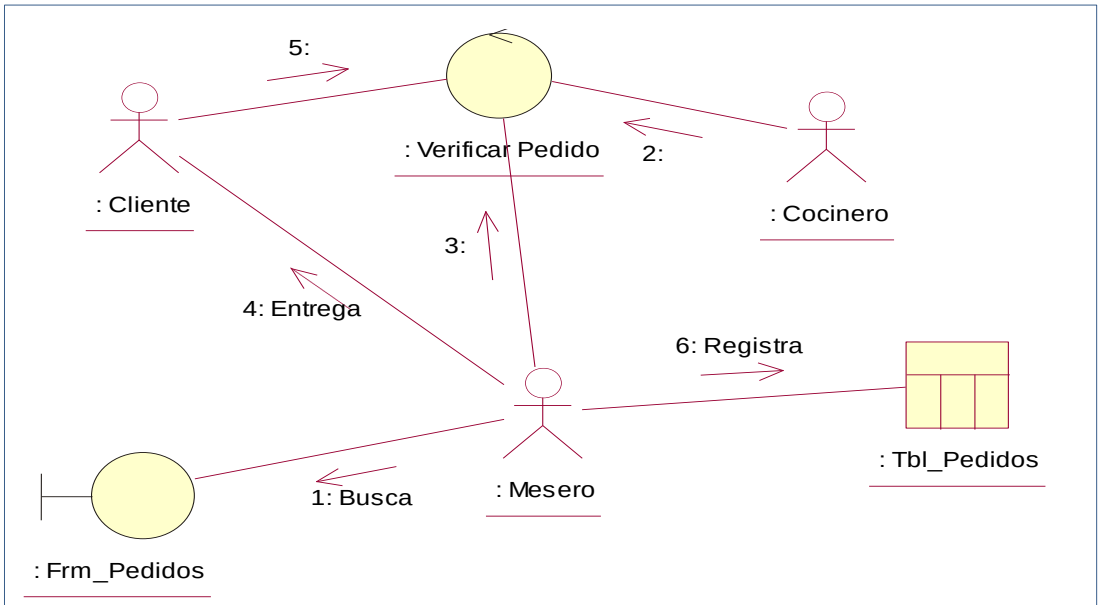


Figura 15. Diagrama de colaboración entregar pedido.

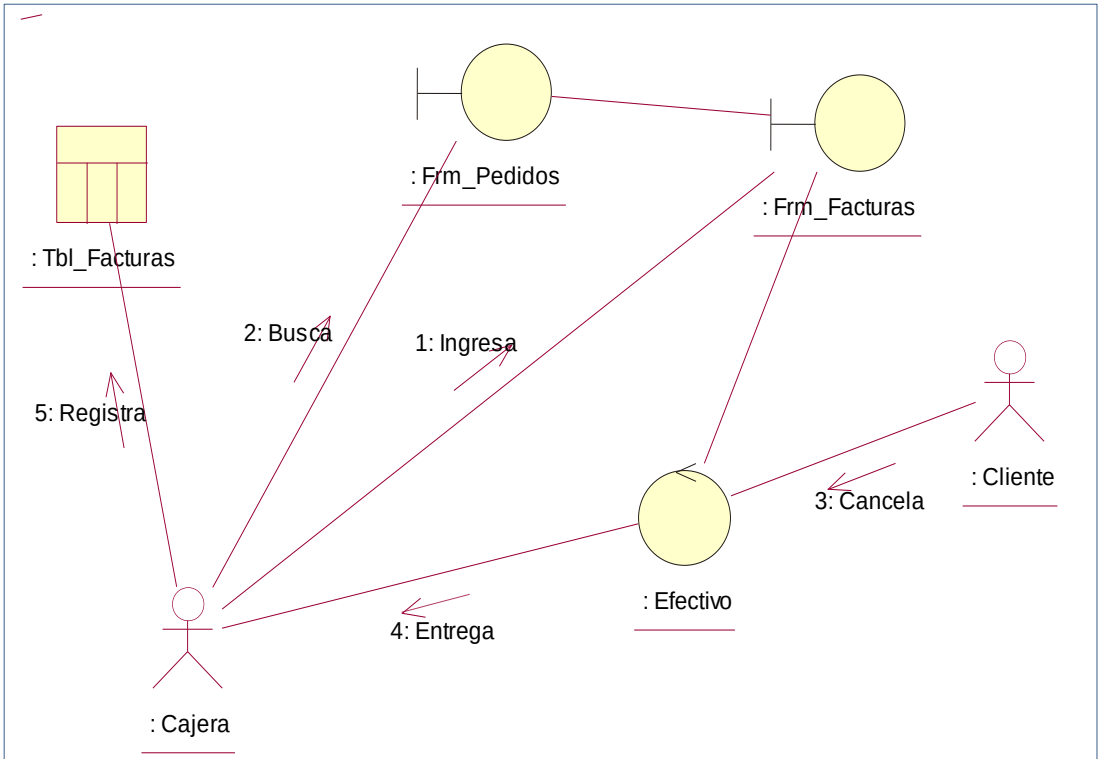


Figura 16. Diagrama de colaboración cancelar pedido.

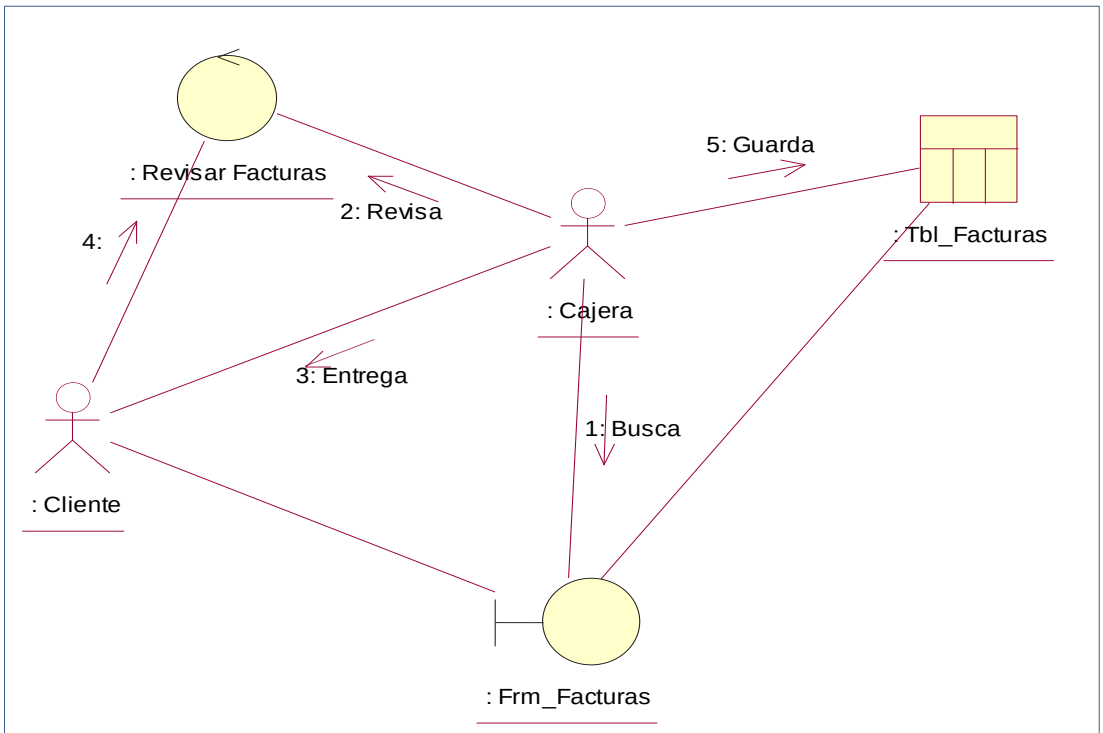


Figura 17. Diagrama de colaboración cancelar pedido.

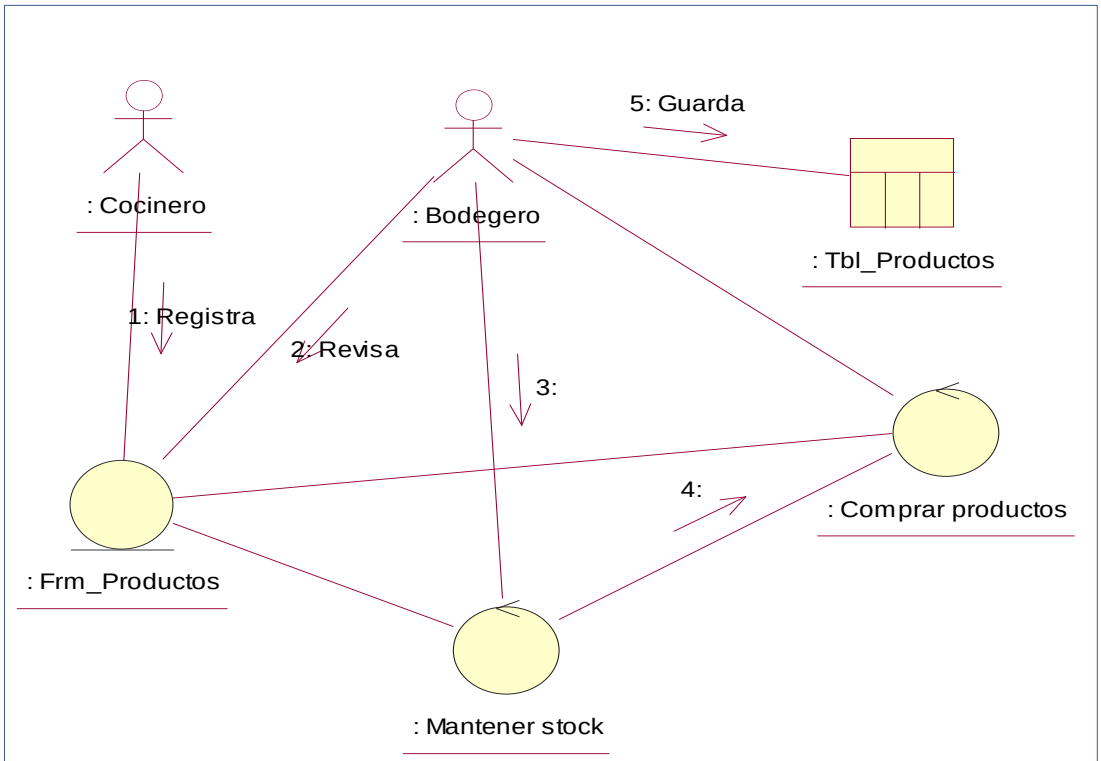


Figura 18. Diagrama de colaboración revisar productos.

5.02.04 Diagrama de componentes

Un componente es una parte física de un sistema (modulo, base de datos, programa ejecutable, etc.). Se puede decir que un componente es la materialización de una o más clases, porque una abstracción con atributos y métodos pueden ser implementados en los componentes.

Figura 19. En un DC, un componente se representa con un rectángulo en el que se escribe su nombre y en él se muestran dos pequeños rectángulos al lado izquierdo.

Este diagrama, también conocido como diagrama de estructura compuesta, tiene por objetivo principal describir con precisión objetos compuestos. Estos diagramas no sustituyen a los diagramas de clases, sino que los completan. En el diagrama de componentes el objeto compuesto se describe mediante un clasificador, mientras que sus componentes se describen mediante las partes. Un clasificador y una parte están asociados a una clase, cuya descripción completa se realiza en un diagrama de clases.

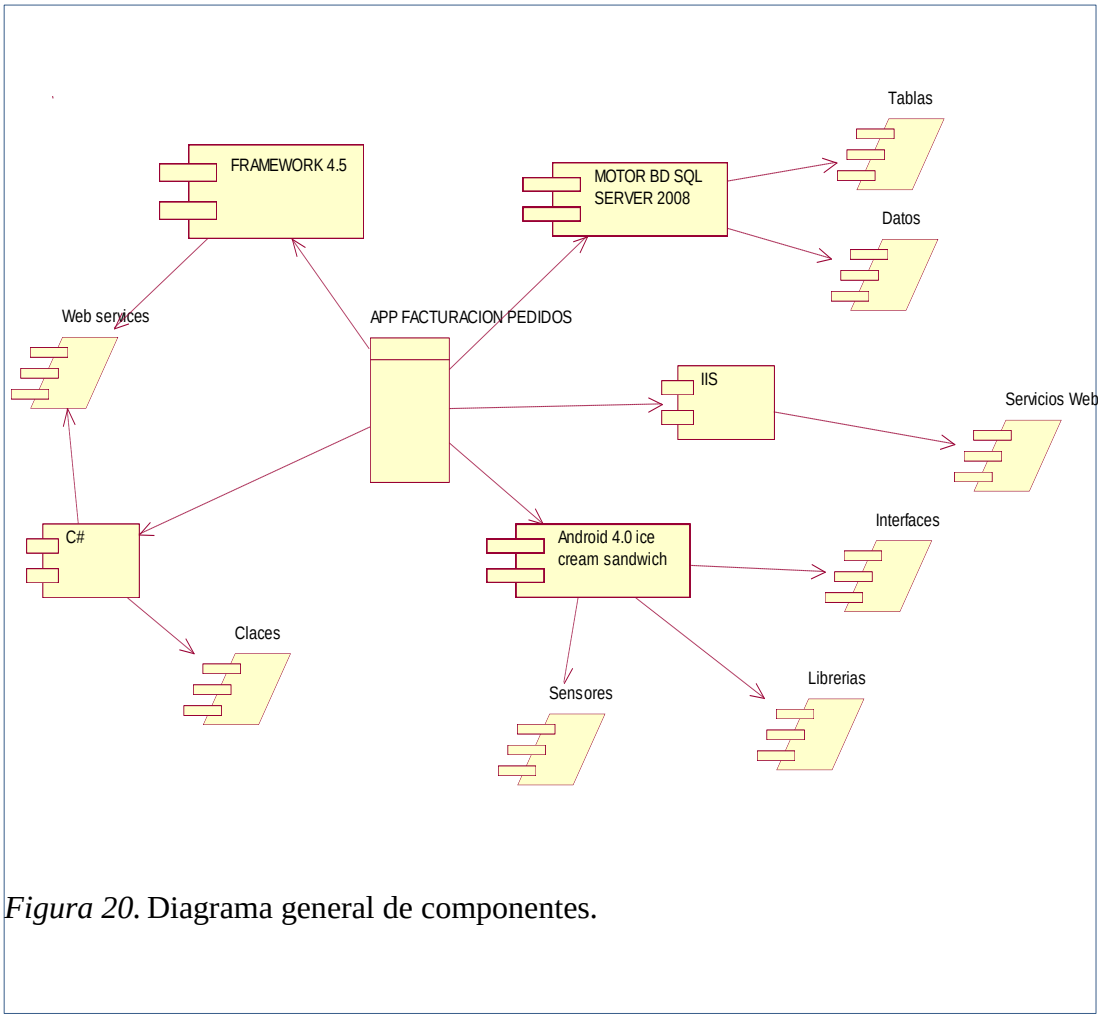


Figura 20. Diagrama general de componentes.

5.02.05 Diagrama físico y lógico.

El diseño lógico de la base de datos es el proceso que determina la estructura lógica de datos necesaria para soportar los recursos de información en una organización. Este proceso de diseño lógico ayuda a implementar la base de datos para satisfacer los requerimientos del usuario. En esta fase se aplica un diseño lógico, el cual esta soportado por medio del análisis relacional de datos o normalización. El análisis relacional de datos permite empezar por definir elementos de datos usados por la organización, para luego normalizar las relaciones en que ellos participan y finalmente obtener un nuevo diagrama lógico de datos (diagrama entidad-relación), el cual

validándolo con el obtenido en fase de análisis, servirá para generar el diseño físico de la base de datos.

El diseño físico es el proceso de traducción del modelo lógico abstracto a un diseño técnico específico para el nuevo sistema. Produce las especificaciones reales para el hardware, software y bases de datos físicas, medios de entrada/salida, procedimientos manuales y controles específicos. Proporciona las especificaciones que transforman el diseño lógico abstracto en un sistema de funciones de personas y máquinas.

Entonces el Diseño Físico de la Base de Datos: Es el proceso de producción de una descripción, de una implementación, de un almacenamiento secundario de la Base de Datos, describe el almacenamiento de estructuras y métodos de acceso usados para conseguir el acceso eficiente a los datos.

Diagrama lógico de la base de datos. (Ver anexo A.06).

Diagrama físico de la base de datos. (Ver anexo A.07).

5.03 Desarrollo

5.03.01 Arquitectura de software.

5.03.01.01. Arquitectura basada en capas.

La arquitectura basada en capas se enfoca en la distribución de roles y responsabilidades de forma jerárquica proveyendo una forma muy efectiva de separación de responsabilidades. El rol indica el modo y tipo de interacción con otras capas, y la responsabilidad indica la funcionalidad que está siendo desarrollada.

Además, permite distribuir el trabajo de creación de una aplicación por niveles; de este modo, cada grupo de trabajo está totalmente abstraído del resto de niveles, de forma que basta con conocer la API que existe entre niveles.

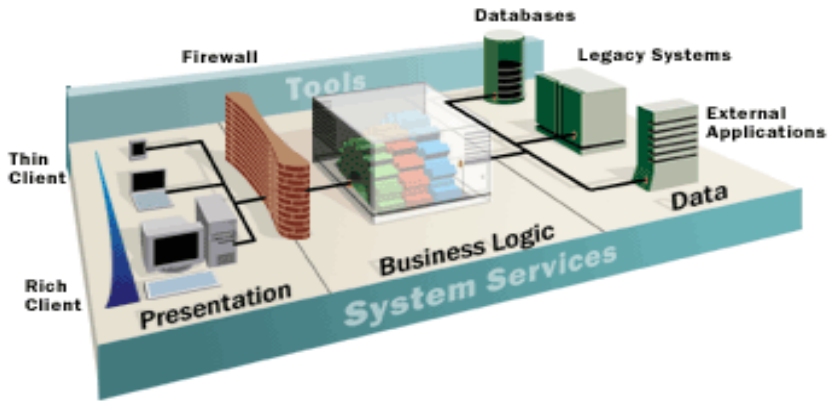


Figura 21. La ventaja principal de este estilo es que el desarrollo se puede llevar a cabo en varios niveles y, en caso de que sobrevenga algún cambio, sólo se ataca al nivel requerido sin tener que revisar entre código mezclado. Un buen ejemplo de este método de programación sería el modelo de interconexión de sistemas abiertos.

5.03.01.02. Arquitectura de N-Capas / 3-Capas.

Este estilo de despliegue arquitectónico describe la separación de la funcionalidad en segmentos separados de forma muy parecida al estilo de capas, pero en el cual cada segmento está localizado en un computador físicamente separado. Este estilo ha evolucionado desde la aproximación basada en componentes generalmente usando métodos específicos de comunicación asociados a una plataforma en vez de la aproximación basada en mensajes.

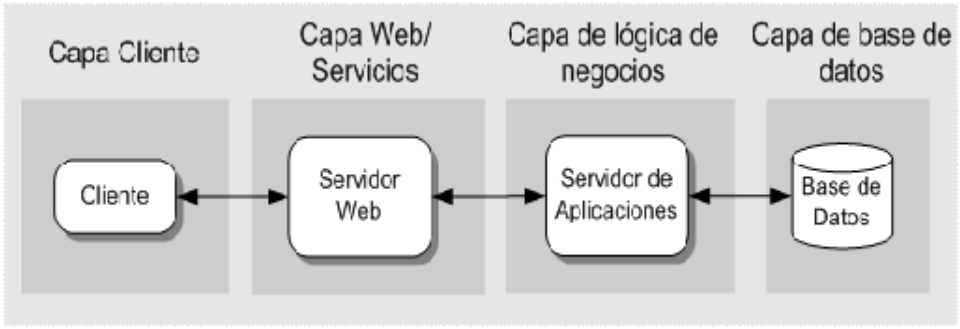


Figura 22. Cada capa es completamente independiente de las otras capas, excepto aquella que esta inmediatamente debajo de ella. La capa n solo necesita saber cómo manejar una solicitud de la capa n+1, como hacer la solicitud a la capa n-1 (si existe) y cómo manejar el resultado de la petición. La arquitectura de N-capas tiene al menos tres capas separadas o partes, cada una de ellas con su responsabilidad y está localizada en diferentes servidores.

5.03.01.03. Arquitectura de Android.

Android está formada por varias capas que facilitan al desarrollador la creación de aplicaciones. Además, esta distribución permite acceder a las capas más bajas mediante el uso de librerías para que así el desarrollador no tenga que programar a bajo nivel las funcionalidades necesarias para que una aplicación haga uso de los componentes de hardware de los teléfonos.

.

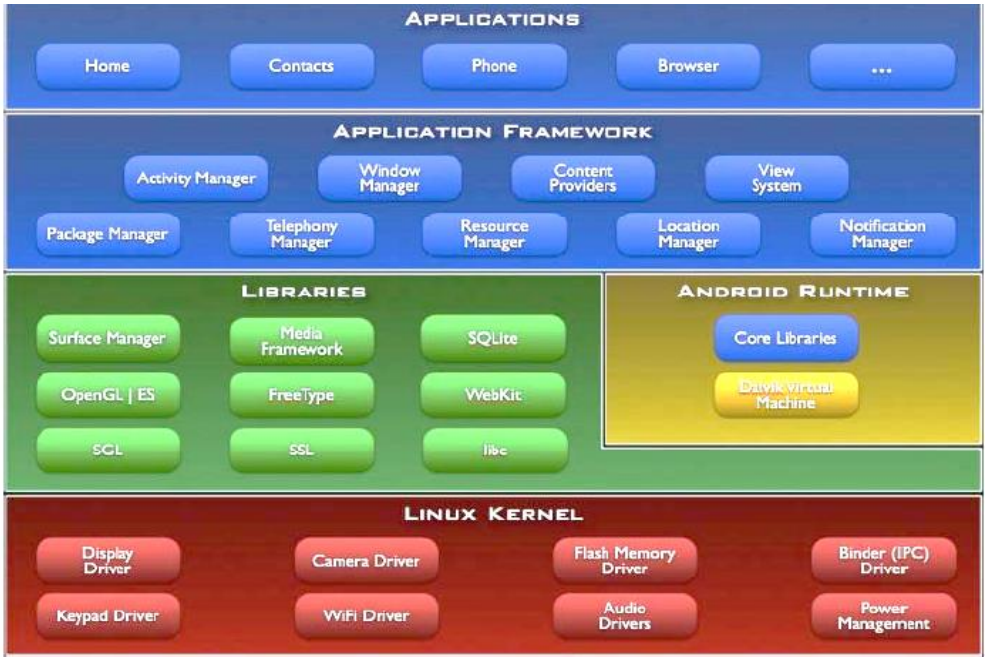


Figura 23. Cada una de las capas utiliza elementos de la capa inferior para realizar sus funciones, es por ello que a este tipo de arquitectura se le conoce también como pila. Para entender mejor, a continuación cito el diagrama de la arquitectura de Android tomada del sitio oficial de Android developers.

5.03.01.04. Requerimientos de software.

- Windows 7 Profesional o Windows 8 Pro, con framework 4.0 y Internet Information Services (IIS) 6.0 ó superior.
- Windows Server 2008 o Windows Server 2012, con framework 4.0 y Internet Information Services (IIS) 6.0 ó superior.
- Android 4.0 Ice Cream Sandwich o superior.

5.03.01.05. Requerimientos de hardware.

- Servidor básico HP ProLiant ML310e Gen 8 E3-1220v2 1P, 2 GB-U, conexión en caliente, 4 LFF, 460 W, PS.
- Tablet Sansung Galaxy Tab 2 7.0 WiFi P3100 Android 4.0, resolución 1.024 x 600 (WSVGA) o superior.
- Computador Intel core I3 3.3GHZ Hdd750 4gb Pantalla Led19.
- Conexión Wireles a internet mínimo un megabit por segundo.

5.03.02 Estándares de base de datos.

La base de datos o schema.

La base de datos SQL Server o los schemas deberán nombrarse usando la siguiente nomenclatura:

- BD_Nombre Base Datos. Ejemplo: BD_FacturacionPedidos.

Tablas.

Las tablas deben nombrarse:

- Tbl_Nombre, en plural, en español y sin utilizar espacios en blanco

Columnas.

Los campos de una tabla corresponden a los atributos de una entidad, describen propiedades de la misma.

Las columnas deben ser nombradas según los lineamientos a continuación:

- Los nombres deben ser simples, representativos e intuitivos.
- Los nombres de las columnas de una tabla deben estar expresados en singular.
- El campo clave de una tabla de nombrarse como el nombre de la tabla más el sufijo Codigo..

Triggers.

Un trigger no tiene sentido fuera de una tabla y un trigger tiene asociada siempre una operación, por lo que dicha información debe estar asociada al nombre del trigger. <Tabla>_<operacion>

Vistas.

Las vistas deben nombrarse con la misma notación definida para nombrar tablas, pero prefijadas usando VW_.

Primary keys.

La clave primaria es un conjunto de campos que identifica de forma única un registro en una tabla. Son un caso particular de un índice. La nomenclatura es la siguiente: pk_<tabla>

Foreign keys.

Como patrón para la nomenclatura de la foreign key elegimos el siguiente.

Fk_<tabla_que_referencia>+<campo_que_referencia>_<tabla_referenciada>+<campo_referenciado>

Indexes.

Los índices están asociados a una tabla y a un conjunto de campos de la tabla, a su vez pueden ser únicos o no y pueden estar definidos en cluster o no. La nomenclatura elegida para nombrarlos es la siguiente: [idx_]<tabla>_<campo>

Tabla 9.

Estándares de bases de datos.

TIPO	PREFIJO	EJEMPLO
Base Datos	BD_	BD_FacturacionPedidos
Tablas	Tbl_	Tbl_Usuario
Campos	Tabla_Nombre	Usu_Codigo
Vistas	VW_	VW_Usuario
Triggers	TablaOperacion	UsuarioEliminar
Primary keys	PK_	PK_Usuario
Foreign keys	FK_	FK_UsuarioIdFacturaId
Indexes	IDX_	IDX_UsuarioNombre

5.03.03 Estándares de programación.

Para implementar estándares de programación, no existe "terminología" o "estilo" que sea mejor que otro. La valoración de dichas "terminologías" se basa no en lo que al programador le guste, sino primordialmente en el uso adecuado de un "terminología" específica. Esto es lo que denominamos "estándares de programación", que no es más que el usar y seguir ciertas reglas de notación y nomenclatura durante la fase de implementación (codificación) de una aplicación.

- No utilizar mayúsculas y minúsculas para diferenciar identificadores.
- Utilizar abreviaturas sólo cuando el nombre completo sea demasiado extenso.
- Nombrar los identificadores de acuerdo a su significado y no a su tipo.
- Evitar el uso de las letras I O y los números 1 y 0 juntos.
- Utilizar verbos para nombrar eventos.
- No incluir el nombre de la clase dentro del nombre de las propiedades.
- Utilizar "//" para los bloques de comentarios.
- Inicializar las variables en el punto de su declaración.
- Verificar la validez de los parámetros que se pasan como argumentos.
- No crear líneas de código de más de 80 caracteres

Tabla 10.

Estándares de Web Services.

TIPO	PREFIJO	EJEMPLO
Web Services	Nombre	Clientes.asmx
Claces de Logica	Logica_	Logica_Cliente.cs
Claces de Modelos	Listar	ListarClientes.cs
Dbml Acceso a Datos	DB	DBAccesoDatos.dbml
Instanciar Clace	Info	InfoCliente
Metodos	TablaInfo	ListarClientesInfo
Retorno gets,sets	Info	InfoCodigo

Estándares de Android 4.0.

- Creación de diferentes layouts para una aplicación Android.
- Diferentes archivos de strings.xml para el soporte de varios idiomas
- Creación de diferentes Bitmaps
- Los iconos deben tener un diseño neutral, plano y simple.
- Los colores deben ser no-neutros con moderación y con un propósito
- Los iconos para notificaciones deben ser de 24×24 dp.
- La resolución de las ventanas debe ser de 1204x600.
- Usar patrones de diseño en Android.

Tabla 11.

Estándares de Android.

TIPO	PREFIJO	EJEMPLO
Layout	Nombre	Cliente.xml
Clace Atividad	Activity	ClientesActivity.java
Clace Datos	ServicioWeb	AccesoServicioWeb.java
Clace Adaptador	Adaptador	AdaptadorCliente.java
Clace get,sets	Listar	ListarClientes.java
ImageButton	imgBtn	imgBtnIngresar
EditText	txt	txtUsuario
TextView	txtw	txtwClientes
Button	btn	btnAceptar
ListView	Lst_	Lst_Clientes

5.03.04 Diseño de interfaces.

Módulo de seguridad.

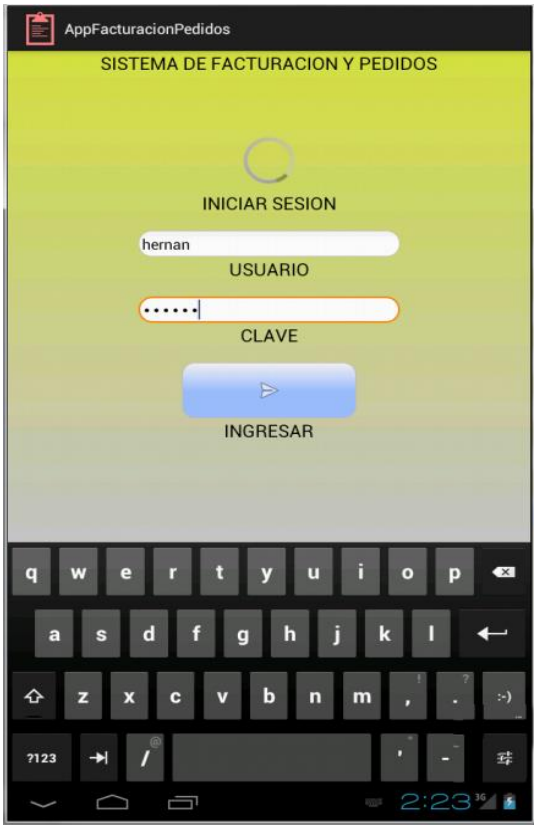


Figura 24. Permite el ingreso de usuarios al sistema validando su usuario, clave y tipo de usuario. Si no se encuentra registrado tiene que acercarse donde el administrador.

Módulo de administración.

Figura 25. Permite al administrador acceder a revisar los reportes o realizar mantenimientos de las tablas de la base de datos

Módulo del usuario.

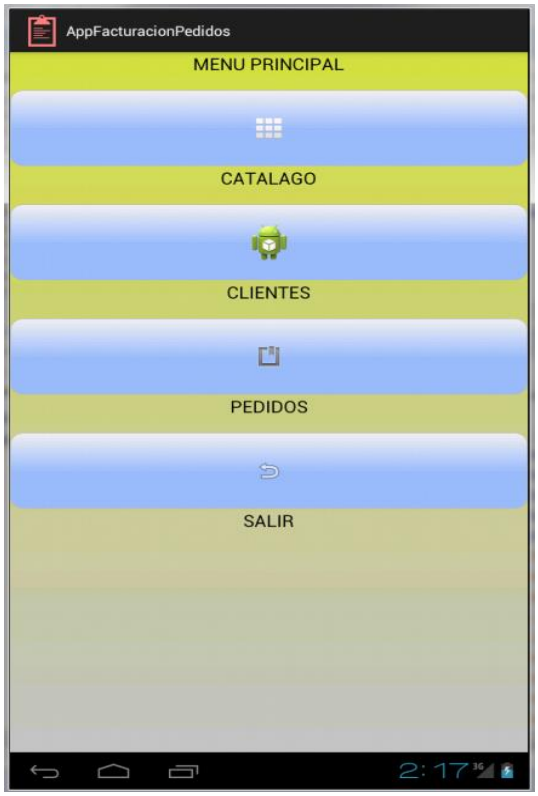


Figura 26 .Este módulo permite a los usuarios, realizar pedidos, revisar facturas, ver el catálogo y el stock de productos.

Lista de Clientes.

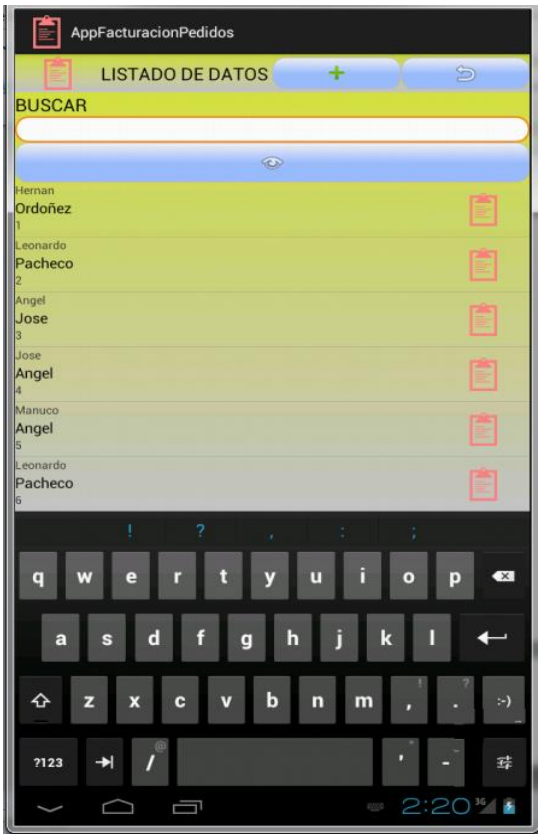


Figura 27. El módulo de clientes, muestra una lista de todos los clientes, detallando el nombre, apellido y código, además haciendo clic sobre cada uno de la lista, se muestra una descripción detallada. También permite realizar la búsqueda por el apellido, agregar un nuevo cliente y recargar los datos.

Registrar o editar un nuevo cliente.

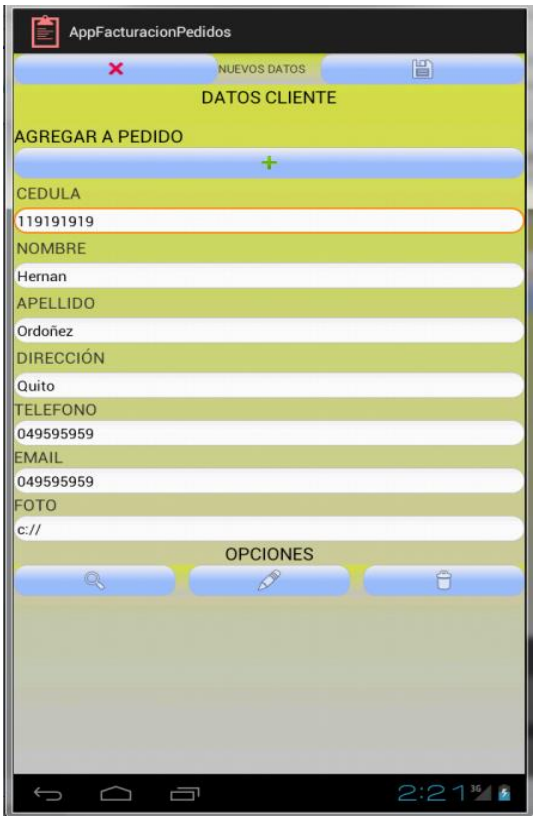


Figura 28. Permite agregar un nuevo cliente a la base de datos, a la vez permite editar y eliminar los datos seleccionados desde la lista de clientes.

5.04 Pruebas

Las pruebas de software es un conjunto de herramientas, técnicas y métodos que hacen a la excelencia del desempeño de un programa. Las técnicas para encontrar problemas en un programa son extensamente variadas y van desde el uso del ingenio por parte del personal de prueba hasta herramientas automatizadas que ayudan a aliviar el peso y el costo de tiempo de esta actividad.

5.04.01 Casos de pruebas

Tabla 12.

Caso de prueba realizar pedido.

Caso de prueba				
Caso de uso	DCR-001. Realizar pedido			
Caso de prueba	CP-001- Realizar pedido			
Actor	Mesero			
Pre Condiciones	El usuario debe iniciar sesión en el sistema			
Propósito	Registrar un pedido			
Escenario	CP-001_E01: Comprobar el correcto ingreso de datos			
	Sec.	**Actividad**	**Clase de equivalencia**	**Resultado esperado**
	1	Mesero ingrese los datos: Cliente y mesa	Válida	Se visualiza los datos ingresados en los respectivos campos
	2	Solicita registrar pedido	Válida	Presenta mensaje realizado correctamente
Escenario	CP-001_E02: Comprobar mensaje de error al guardar con datos vacíos			
	1	Solicita registrar pedido	Válida	Presenta mensaje los datos no pueden estar vacíos
Escenario	CP-001_E03: Comprobar mensaje de cancelación de registro			
	1	Solicita registrar pedido	Válida	Presenta mensaje ok, cancelar

Tabla 13.

Caso de prueba entregar pedido.

Caso de prueba				
Caso de uso	DCR-002. Entregar pedido			
Caso de prueba	CP-002- Entregar pedido			
Actor	Mesero			
Pre Condiciones	El pedido debe estar registrado en el sistema			
Propósito	Entregar el pedido al cliente			
Escenario	CP-002_E01: Comprobar el correcto ingreso de datos			
	Sec.	**Actividad**	**Clase de equivalencia**	**Resultado esperado**
	1	Mesero ingrese los datos: Mesa, Cliente	Válida	Se visualiza el detalle de los pedidos
	2	Solicita registrar entrega	Válida	Presenta mensaje realizado correctamente
Escenario	CP-002_E02: Comprobar mensaje de error al guardar con datos vacíos			
	1	Solicita registrar entrega	Válida	Presenta mensaje los datos no pueden estar vacíos
Escenario	CP-002_E03: Comprobar mensaje de cancelación de registro			
	1	Solicita registrar entrega	Válida	Presenta mensaje ok, cancelar

Tabla 14.

Caso de prueba cancelar pedido.

Caso de prueba				
Caso de uso	DCR-003. Cancelar pedido			
Caso de prueba	CP-003- Cancelar pedido			
Actor	Cajera			
Pre Condiciones	El pedido debe estar registrado en el sistema			
Propósito	Registrar como cancelado el pedido			
Escenario	CP-003_E01: Comprobar el correcto ingreso de datos			
	Sec.	**Actividad**	**Clase de equivalencia**	**Resultado esperado**
	1	Cajera ingrese los datos: Mesa, Cliente	Válida	Se visualiza la factura
	2	Solicita registrar pago	Válida	Presenta mensaje realizado correctamente
Escenario	CP-003_E02: Comprobar mensaje de error al guardar con datos vacíos			
	1	Solicita registrar pago	Válida	Presenta mensaje los datos no pueden estar vacíos
Escenario	CP-003_E03: Comprobar mensaje de cancelación de registro			
	1	Solicita registrar pago	Válida	Presenta mensaje ok, cancelar

Tabla 15.

Caso de prueba entregar factura.

Caso de prueba				
Caso de uso	DCR-004. Entregar factura			
Caso de prueba	CP-004- Entregar factura			
Actor	Cajera			
Pre Condiciones	Debe haber registro de pago de la factura			
Propósito	Registrar entregar de factura			
Escenario	CP-004_E01: Comprobar el correcto ingreso de datos			
	Sec.	**Actividad**	**Clase de equivalencia**	**Resultado esperado**
	1	Cajera ingrese los datos: Mesa, Cliente	Válida	Se visualiza el pago de la factura
	2	Solicita imprimir o enviar a mail	Válida	Presenta mensaje realizado correctamente
Escenario	CP-004_E02: Comprobar mensaje de error al generar factura sin datos			
	1	Solicita imprimir o enviar a mail	Válida	Presenta mensaje los datos no pueden estar vacíos
Escenario	CP-004_E03: Comprobar mensaje de cancelación de entrega de factura			
	1	Solicita imprimir o enviar a mail	Válida	Presenta mensaje ok, cancelar

Tabla 16.

Caso de prueba revisar productos.

Caso de prueba				
Caso de uso	DCR-005. Revisar Productos			
Caso de prueba	CP-005- Revisar Productos			
Actor	Bodeguero- Cocinero			
Pre Condiciones	Debe registrarse el consumo de productos			
Propósito	Registrar el stock de productos			
Escenario	CP-005_E01: Comprobar el correcto ingreso de datos			
	Sec.	**Actividad**	**Clase de equivalencia**	**Resultado esperado**
	1	Cocinero y mesero ingresan los datos: Nombre Stock	Válida	Se visualiza los productos en stock
	2	Solicita registrar stock	Válida	Presenta mensaje realizado correctamente
Escenario	CP-005_E02: Comprobar mensaje de error al guardar con datos vacíos			
	1	Solicita registrar stock	Válida	Presenta mensaje los datos no pueden estar vacíos
Escenario	CP-005_E03: Comprobar mensaje de cancelación de registro			
	1	Solicita registrar stock	Válida	Presenta mensaje ok, cancelar

5.04.02 Matriz de control de pruebas

Tabla 17.

Matriz de control de pruebas.

Proceso	Acción	Estado	Observación
Inicio Sesión	Verifica si usuario está registrado	Bueno	Falta validar los intentos de inicio de sesión.
Registrar Clientes	Añadir nuevo cliente a la Base Datos	Bueno	Validación de cédula no es correcta.
Facturación	Mesero genera factura	Bueno	El correo electrónico no se valida.
Registro nuevo usuario	Guardar BD usuario	Bueno	Validar ya existencia de usuario y clave.
Registrar Pedidos	Registrar pedido en BD	Bueno	Falta validar que no se envíe a guardar sin datos.
Consultar Reportes	Ver reportes del sistema	Bueno	Validar que parámetros del reporte no queden vacíos.
Realizar factura	Generar factura	Bueno	Validar que no se envíe a generar la factura sin ítems.
Registrar productos	Guardar productos en BD	Bueno	Validar el ingreso correcto del formato de precios.

5.04.03 Parametrización del software

El software para ser flexible, posee una gran cantidad de parámetros que permiten que el administrador del sistema pueda adaptar el software de acuerdo a las normas vigentes y además tener la información clasificada.

- Administradores del sistema.
- Usuarios del sistema
- Productos terminados que se encuentran en stock
- Productos que se encuentran en bodega
- Tipos de impuestos y localización geográfica.
- Vigencias y períodos de facturación anual, bimensual, semestral o con la regularidad que exija el impuesto asociado.
- Parámetros requeridos para cada uno de los impuestos.
- Parámetros asociados a las vigencias y períodos de facturación.
- Formas de pago.
- Descuentos por impuestos, vigencia y período de facturación.
- Habilitar reservas en pedidos
- Número de decimales a trabajar.
- Permitir Margen Negativo
- Número de copias por defecto para facturas

5.04.04 Matriz de errores

Los errores descritos en la siguiente matriz muestran las posibles situaciones externas que pudieran ocurrir, durante el funcionamiento del software, y su posible solución para el correcto funcionamiento.

Tabla 18.

Matriz de errores.

Factores	Error	Status	Solución
Configuración de la fecha	Formato incorrecto	Media	Configurar formato dd/mm/aa
Versión de Android inferior a 4.0	No se puede instalar aplicación	Baja	Actualizar versión a 4.0 o superior
Configuración de red inalámbrica	No se puede conectar al servidor	Media	Activar dhcp en router inalámbrico y tablets
Versión de Framework en servidor	No funciona web services	Baja	Actualizar la versión de Framework a 4.5
Servidor desconectado	No existe respuesta del servidor	Baja	Revisar red y funcionamiento correcto del servidor.

Capítulo VI: Aspectos Administrativos

6.01 Recursos

Capital humano

Para la realización del proyecto han sido necesarias las siguientes personas.

- Equipo de trabajo: Hernán Ordóñez.
- Asesores: Ing. Richard Mafla, Ing. Patricia Garzón.
- Usuarios: Administrador y personal del restaurante Real Manabita.

Recursos Materiales

Los materiales necesarios para la ejecución del proyecto son los siguientes.

- Equipos: Portátil, Tablet, Router Inalámbrico.
- Bibliografía: Tutoriales y manuales de programación de Servicios Web y consumo de los mismos desde Android.
- Copias: Impresión del documento de la realización del proyecto.
- Servicios: Internet Inalámbrico, energía eléctrica.
- Transporte: Pago de pasajes para traslado al ITSCO.

Recursos Económicos

El costo total del proyecto es de $1200 dólares, tomando como referencia, las facturas de los equipos y recursos necesarios para la ejecución del mismo.

6.02 Presupuesto

Ingresos

Aporte personal…….1200

EGRESOS	**VALOR**
Elaboración del proyecto …….…….…….	350
Material de escritorio……...…..	50
Copias…….……....	50
Adquisición de equipos……...…..…..	500
Gastos administrativos……..….....	100
Transporte…….……	50
Imprevistos……..……..	100

TOTAL 1200

6.03 Cronograma

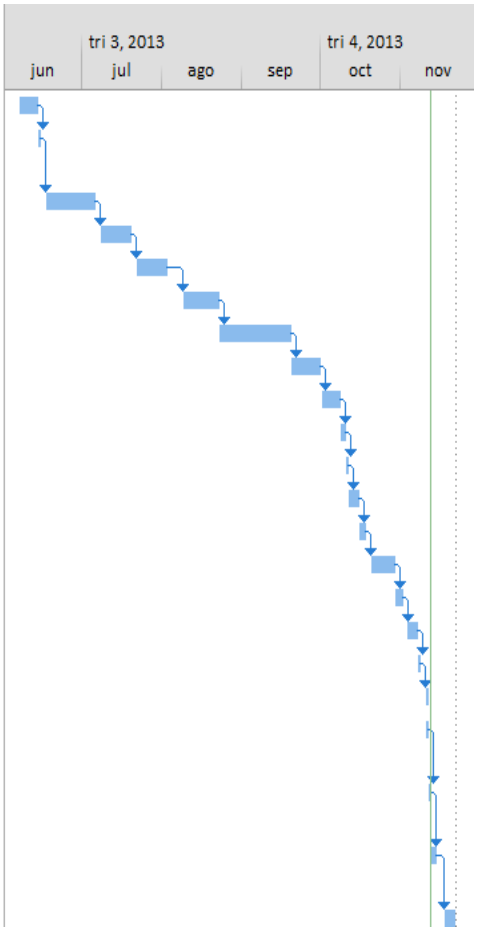
El siguiente cronograma corresponde a la programación de las actividades básicas del proyecto y la distribución del tiempo estimado para su cumplimiento.

Tabla 19.

Cronograma.

DIAGRAMA DE GANTT

	Nombre de tarea	Duraci	Comienzo	Fin
1	Aprobacion del Tema	5 días	vie 07/06/13	jue 13/06/13
2	Designacion de tutor y lector	1 día	vie 14/06/13	vie 14/06/13
3	Capítulo I	15 días	lun 17/06/13	vie 05/07/13
4	Capítulo II	10 días	lun 08/07/13	vie 19/07/13
5	Capítulo III	10 días	lun 22/07/13	vie 02/08/13
6	Capítulo IV	10 días	vie 09/08/13	jue 22/08/13
7	Capítulo V	20 días	vie 23/08/13	jue 19/09/13
8	Capítulo VI	7 días	vie 20/09/13	lun 30/09/13
9	Capitulo VII	5 días	mié 02/10/13	mar 08/10/13
10	Anexos	2 días	mié 09/10/13	jue 10/10/13
11	Indices	1 día	vie 11/10/13	vie 11/10/13
12	Arreglos Finales	3 días	sáb 12/10/13	mar 15/10/13
13	Revision de Tutor	3 días	mié 16/10/13	vie 18/10/13
14	Correcciones	7 días	lun 21/10/13	mar 29/10/13
15	Presentación al tutor	3 días	mié 30/10/13	vie 01/11/13
16	Presentación al lector	4 días	lun 04/11/13	jue 07/11/13
17	Correcciones	1 día	vie 08/11/13	vie 08/11/13
18	Aprobacion de Lector	1 día?	lun 11/11/13	lun 11/11/13
19	Presentacion al Director de escuela	1 día?	lun 11/11/13	lun 11/11/13
20	Entrega de Acta de Aprobacion	1 día?	mar 12/11/13	mar 12/11/13
21	Presentación de Anillados y empastado	2 días	mié 13/11/13	jue 14/11/13
22	Defensas	5 días	lun 18/11/13	vie 22/11/13



Capítulo VII: Conclusiones y Recomendaciones

7.01 Conclusiones

Al finalizar este trabajo de investigación se han podido abstraer las siguientes principales conclusiones:

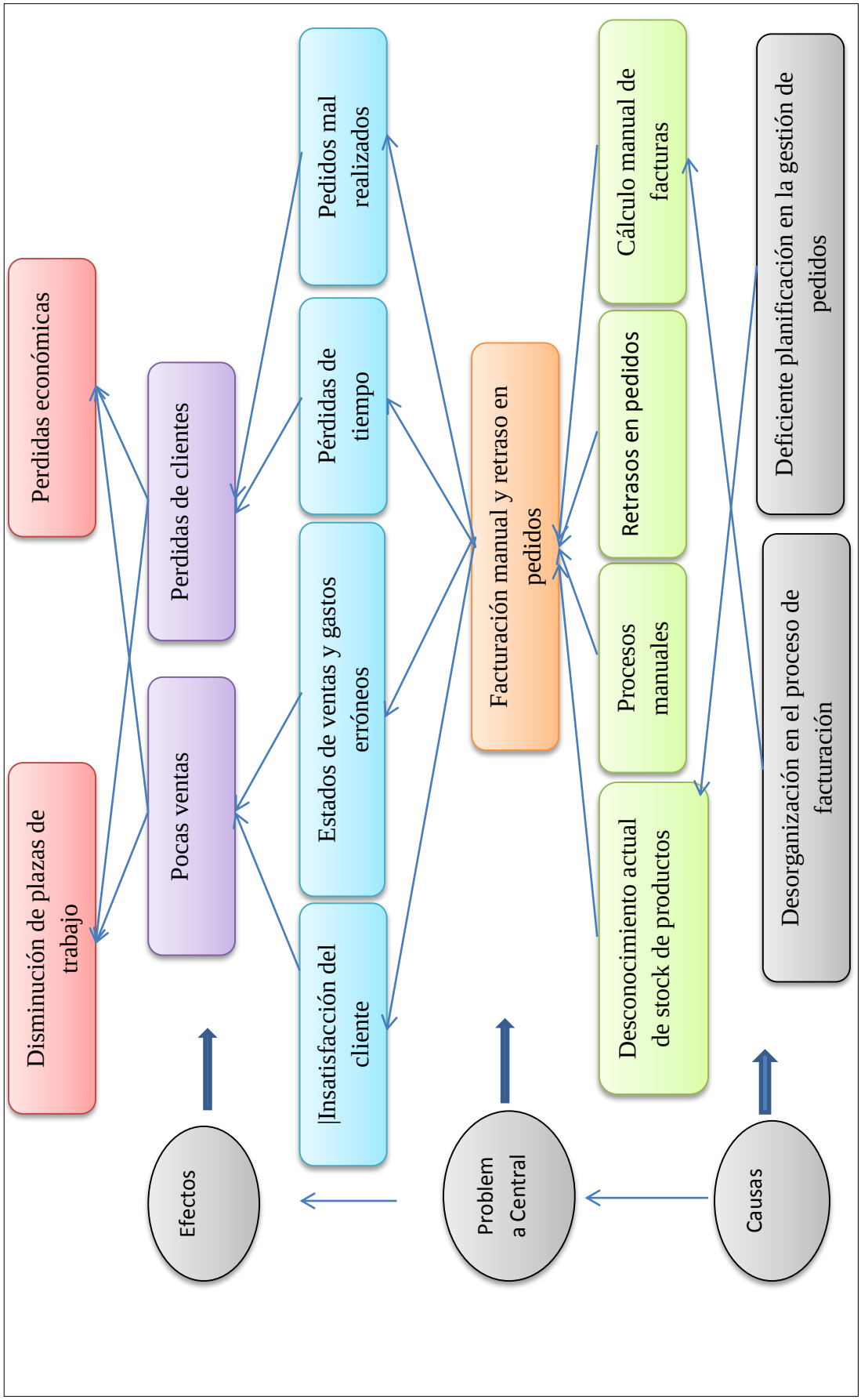
- Se identificaron los procesos de facturación, toma de pedidos y compra de productos del restaurante Real Manabita y en función a ello se hizo el requerimiento para implementar un sistema automatizado.
- Se identificaron las actividades en las cuales se llevan los procesos manuales, para realizar un análisis de la información.
- Se encontró que la facturación no dispone de un entorno tecnológico funcional flexible, los procesos de pago son lentos con errores y no se cumple correctamente los deberes tributarios, lo cual afecta en la imagen de la empresa.
- Se analizó los procesos de la empresa, los cuales no permiten realizar una auditoría de forma ágil y rápida ya que la búsqueda y localización de los documentos, conlleva un largo y tedioso proceso manual.

7.02 Recomendaciones

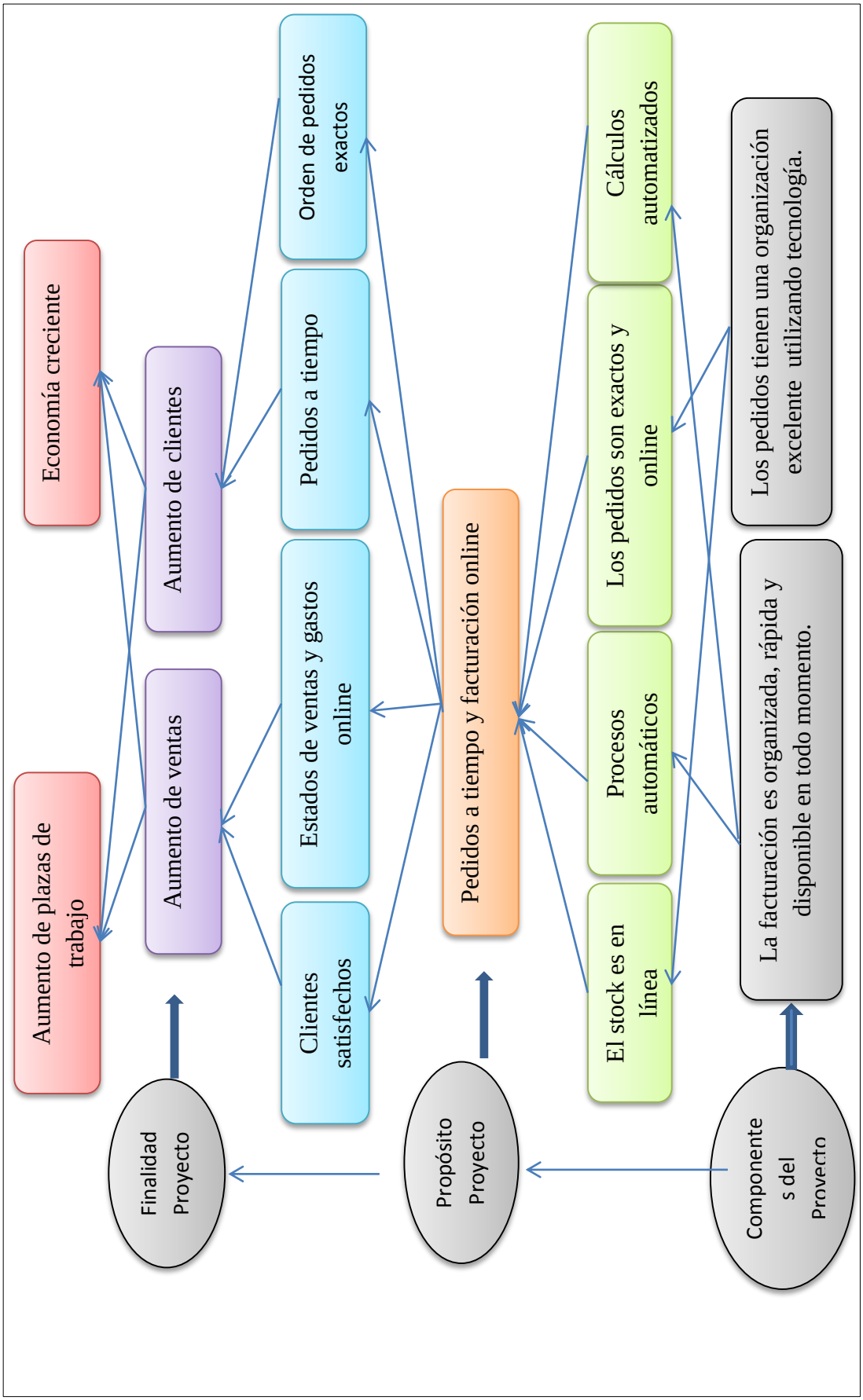
- Realizar la facturación electrónica para que la empresa se modernice y ayude a los procesos de auditoria de la misma.
- Implementar una solución automatización utilizando las tecnologías de la información y la comunicación.
- Sistematizar la información tecnológica de modo que pueda ser amigable con el usuario final, para brindar una mejor atención al cliente
- Debe mantenerse actualizado los productos en la base de datos para poder realizar los pedidos y las facturas de forma oportuna.

ANEXOS

A.01. Árbol de problemas



A.02. Árbol de objetivos



A.03. Matriz de análisis del impacto de los objetivos

Objetivos	Factibilidad *(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
Pedidos a tiempo y facturación online	**20 puntos**	**18 puntos**	**18 puntos**	**18 puntos**	**18 puntos**	**Total**
	--Se cuenta con el debido soporte político de la Empresa. --Los beneficios que se obtiene son mayores que los costos. --Existe la tecnología disponible y se conoce de su funcionamiento para su realización. --Es aceptable y conveniente para el Restaurante Real Manabita. --Cuenta con financiamiento.	--Se fortalece la aplicación de los derechos de las personas. --Incrementa la participación de las mujeres. --Incrementa el nivel educativo del personal. --Incrementa los ingresos de las familias. --Aumenta la autoestima de los empleados.	--Mejora el entorno social. --Mejora el entorno cultural. --Protege el uso de los recursos naturales. --Favorece la educación ambiental. --Enseña a reciclar.	--Responde las expectativas de la empresa --Es una prioridad sentida por los usuarios. --Los beneficios son deseados por los clientes. --Beneficia a los empleados y clientes. --Satisface la expectativa del personal administrativo.	--Se puede conseguir financiamiento a futuro. --Se fortalece la organización de la empresa. --Favorece la participación de los empleados y clientes. --Disponibilidad de información a futuro. --Es un proceso innovador.	92 puntos 20-40 Baja 40-60 Media Baja 60-80 Media Alta 80-100 **Alta**

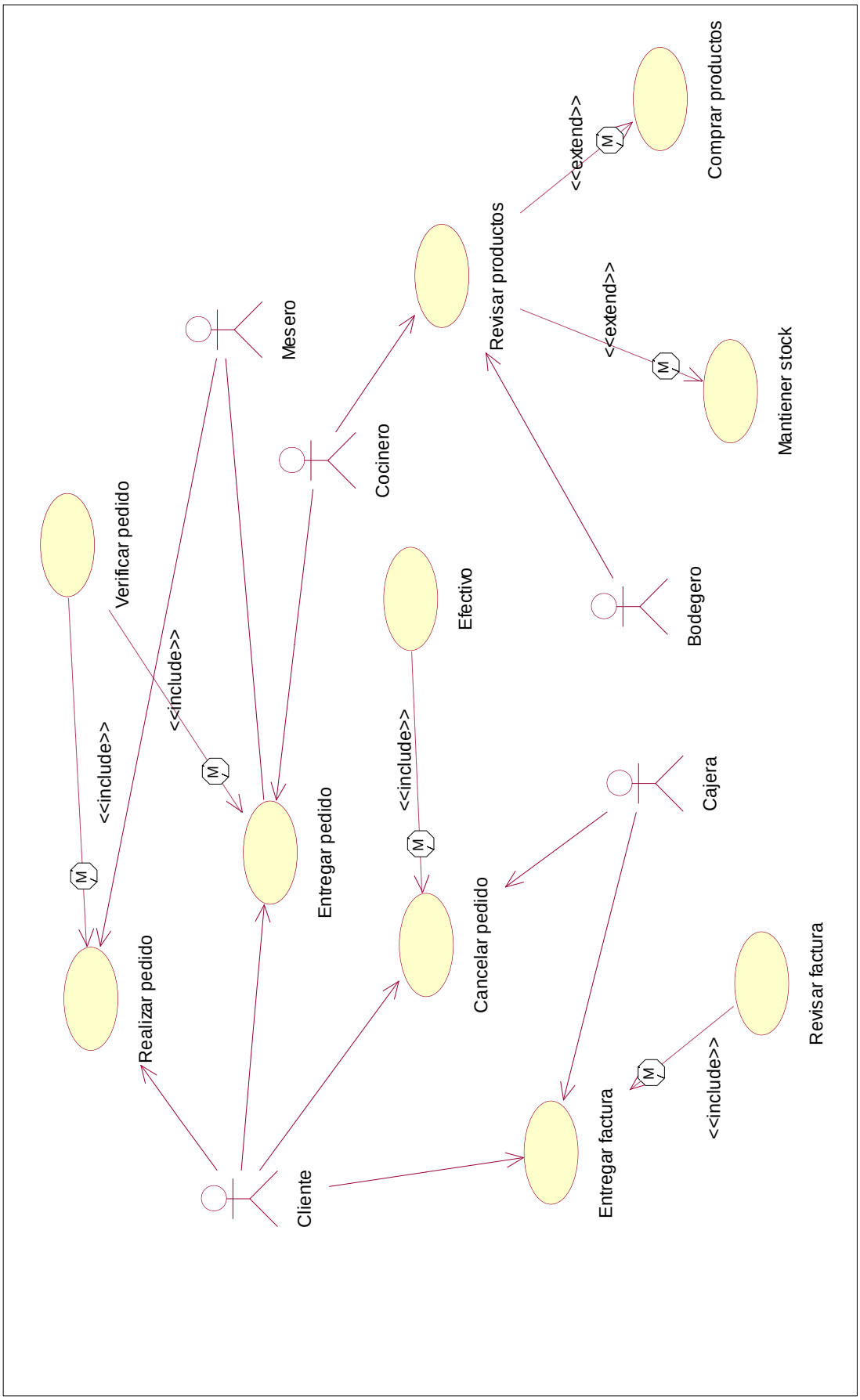
A.04.01. Matriz del marco lógico

RESUMEN NARRATIVO	INDICADORES	MEDIOS DE VERIFICACIÓN	SUPUESTO
FIN DEL PROYECTO Aumentan las ventas y los clientes asisten con mayor frecuencia.	El número de clientes atendidos: 100 en el año 2012, 200 a fines del año 2013, 300 a fines del año 2014, 400 para Diciembre del 2015 y 500 para Diciembre 2016. El número de quejas de clientes disminuye de 200 en el año 2012 a 30 a fines del año 2013, 20 a fines del año 2014, 10 para Diciembre del 2015 y 5 para Diciembre del 2016.	Resultado de estadísticas obtenidas de los comentarios y sugerencias de los clientes. Resultados de encuestas entre clientes y personal del restaurante.	Ampliación del negocio a otros sectores de la ciudad.
PROPÓSITO DEL PROYECTO Los pedidos son a tiempo y la facturación es online	Los errores en facturación disminuyen de 400 en el año 2012 a 40 a fines del año 2013, 20 a fines del año 2014, 10 a fines del año 2015 5 para Diciembre del 2016). El número de retraso en pedidos (+/- 5 minutos) baja de 500 en el año 2012 a 20 a fines del año 2013, 15 a fines del año 2014, 10 a fines del año 2015 y 5 Diciembre de 2016).	Estadísticas obtenidas mediante el sistema de información. Estadísticas obtenidas del nivel de satisfacción de los clientes.	El precio relativo del servicio de internet y la energía eléctrica se mantienen estables.
COMPONENTES DEL PROYECTO 1).La facturación es organizada, rápida y disponible en todo momento. 2).Los pedidos tienen una organización excelente utilizando tecnología de la información.	La desorganización de la facturación disminuye de 500 facturas en el año 2012 a 20 a fines del año 2013, 15 a fines del año 2014, 10 a fines del año 2015 y 5 al Diciembre del 2016. (1) El desorden de los pedidos disminuye de 400 pedidos mal realizados en el año 2012 a 25 a fines del año 2013, a 20 a fines del año 2014, a 15 a fines del año 2015 y a 5 Diciembre del 2016.(2)	Estadísticas obtenidas del resultado de la facturación, mediante el almacenamiento de información, en bases de datos.	Disminución de recursos y tiempo en la facturación y agilidad en la toma de pedidos.

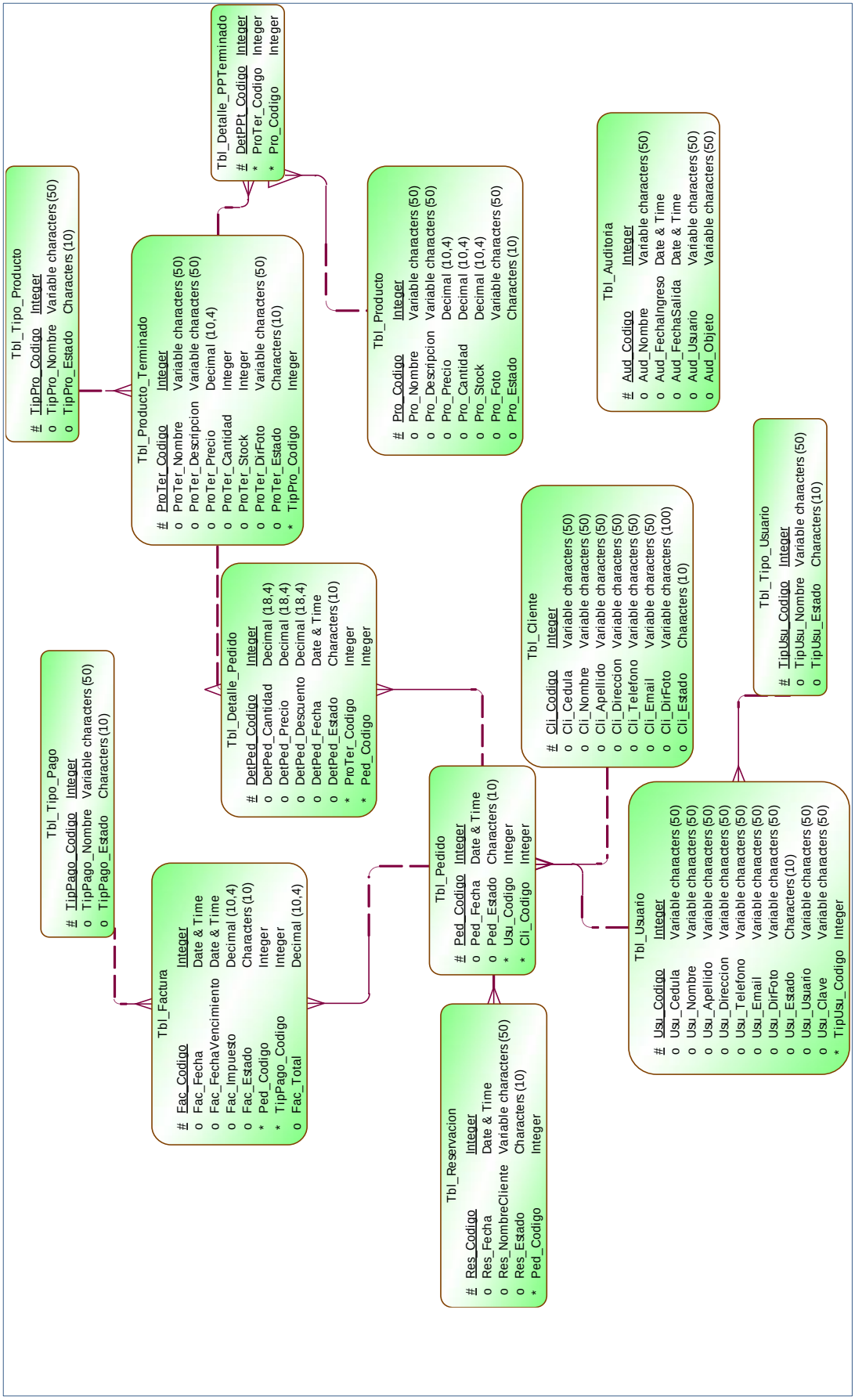
A.04.02. ***Matriz del marco lógico***

ACTIVIDADES DEL PROYECTO	**PRESUPUESTO**		
Implementar los procesos automáticos.	Talento Humano : $400 Dispositivo móvil: $500 Otros Costos(Trasporte, impresiones, internet): $100 Total $1000	Facturas de equipos, internet.	Los costos de las bases de datos y aplicaciones para desarrollo se mantienen estables.
Realizar la automatización de los cálculos en facturas.	Reserva de contingencia: $100 Reserva de gestión: $100		
Implementar pedidos en línea, sin fallas ni errores.	Total de Presupuesto: $1200		
Desarrollar el stock de productos online.			

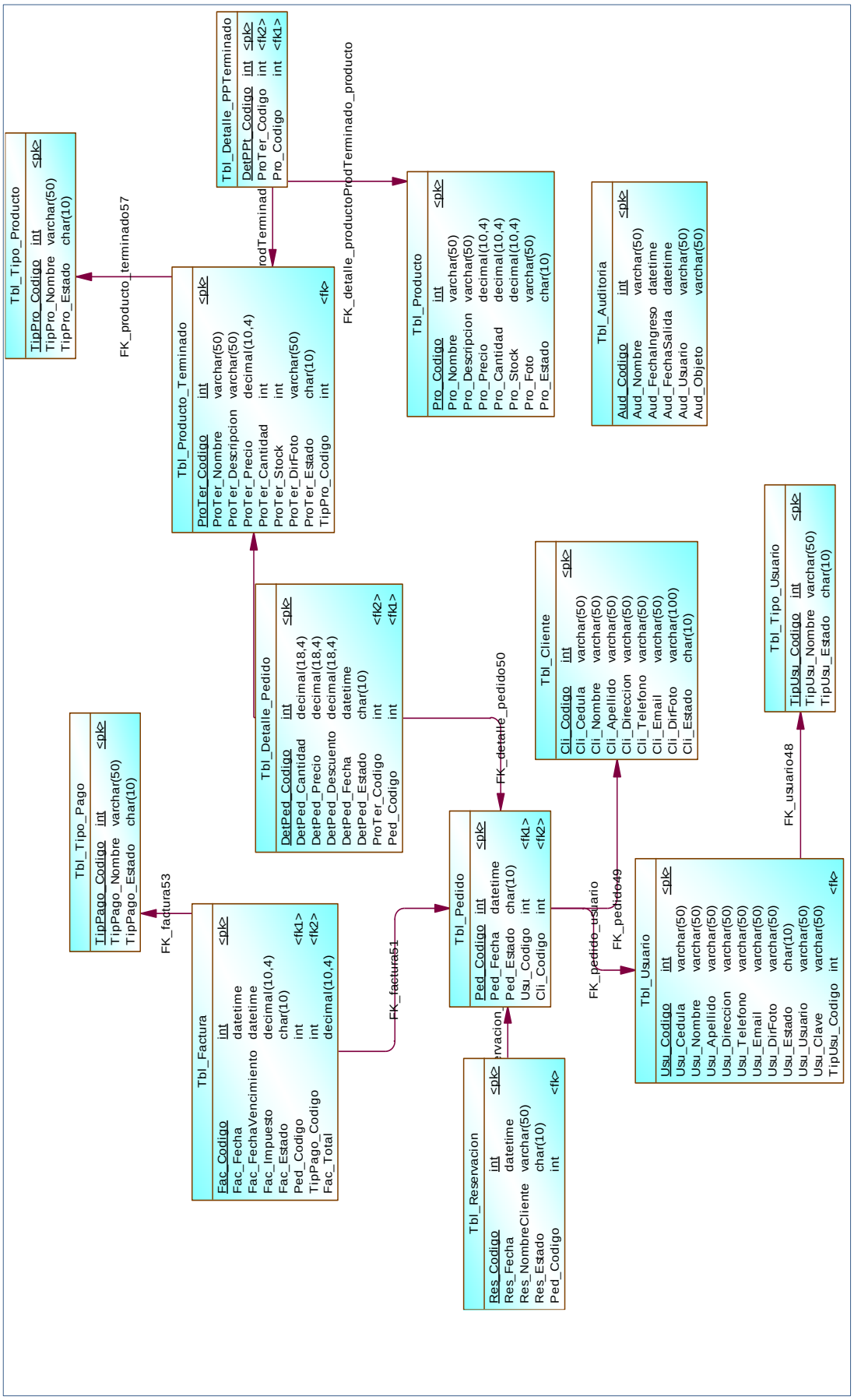
A.05. Diagrama de caso de uso general



A.06. Diagrama lógico de la base de datos



A.07. Diagrama físico de la base de datos



A.08. Diccionario de datos

1. Introducción

En el siguiente diccionario de datos contiene las características lógicas y puntuales de los datos, aquí se encuentra la lista de todos los elementos que forman parte del flujo de datos de todo el sistema.

Tabla 20.

Diccionario de datos.

Tabla	Campo	Tipo dato	Longitud	Llave primaria	Descripción	Tabla foránea
Tbl_Usuario	Usu_Codigo	int	10	PK	Código del usuario	
	Usu_Cedula	varchar	50		Cedula del usuario	
	Usu_Nombre	varchar	50		Nombre del usuario	
	Usu_Apellido	varchar	50		Apellido del usuario	
	Usu_Direccion	varchar	50		Dirección del usuario	
	Usu_Telefono	varchar	50		Teléfono del usuario	
	Usu_Email	varchar	50		Email del usuario	
	Usu_DirFoto	varchar	50		Foto del usuario	
	Usu_Estado	char	10		Estado del usuario	
	Usu_Usuario	varchar	50		Usuario del usuario	
	Usu_Clave	varchar	50		Clave del usuario	
	TipUsu_Codigo	int	10	FK	Código del tipo de usuario	Tbl_Tipo_Usuario
Tbl_Cliente	Cli_Codigo	int	10	PK	Código del cliente	
	Cli _Cedula	varchar	50		Cedula del cliente	
	Cli _Nombre	varchar	50		Nombre del cliente	
	Cli _Apellido	varchar	50		Apellido del cliente	
	Cli _Direccion	varchar	50		Dirección del cliente	
	Cli _Telefono	varchar	50		Teléfono del cliente	
	Cli _Email	varchar	50		Email del cliente	
	Cli _DirFoto	varchar	50		Foto del cliente	
	Cli _Estado	char	10		Estado del cliente	
Tbl_Producto_Terminado	ProTer_Codigo	int	10	PK	Código del producto	
	ProTer_Nombre	varchar	50		Nombre del producto	
	ProTer_Descripcion	varchar	50		Descripción del producto	
	ProTer_Precio	decimal	10, 2		Precio del producto	

	ProTer_Cantidad	int	10		Cantidad de producto	
	ProTer_Stock	int	10		Stock del producto	
	ProTer_DirFoto	varchar	50		Foto del producto	
	ProTer_Estado	char	10		Estado del producto	
	TipPro_Codigo	Int	10	FK	Código del tipo de producto	Tbl_Tipo_Producto
Tbl_Pedido	Ped_Codigo	int	10	PK	Código del pedido	
	Ped_Fecha	datetime			Fecha del pedido	
	Ped_Estado	char	10		Estado del pedido	
	Ped_Estado2	char	10		Estado 2 de pedido	
	Usu_Codigo	int	10	FK	Código del usuario	Tbl_Usuario
	Cli_Codigo	int	10	FK	Código del cliente	Tbl_Cliente
	Mes_Codigo	int	10	FK	Código de la mesa	Tbl_Mesa
Tbl_Factura	Fac_Codigo	int	10	PK	Código de la factura	
	Fac_Fecha	datetime			Fecha de la factura	
	Fac_FechaVencimiento	datetime			Fecha de vencimiento de la factuta	
	Fac_Impuesto	decimal	10, 2		Impuesto de la factura	
	Fac_Estado	char	10		Estado de la factura	
	Ped_Codigo	int	10	FK	Código del pedido	Tbl_Pedido
	TipPago_Codigo	int	10	FK	Código del tipo de pago	Tbl_Tipo_Pago
	Fac_Total	decimal	10, 2		Total de la factura	
Tbl_Mesa	Mes_Codigo	int	10	PK	Código de la mesa	
	Mes_Numero	int	10		Número de la mesa	
	Mes_Descripcion	varchar	50		Descripción de la mesa	
	Mes_Estado	char	10		Estado de la mesa	

A.09. Manual Técnico

1. Introducción

El objetivo del presente manual es mostrar los datos técnicos en cuanto al sistema desarrollado, en si para facilitar la modificación o actualización del mismo en caso de que sea necesario, o bien para el mantenimiento posterior del mismo con el fin de que analistas, programadores pueden leerlo e interpretarlo para los objetivos anteriormente descritos.

2. Diseño de la Base de datos

Para un funcionamiento eficiente de la aplicación se diseñó la Base de Datos en SQL Server 2008, debido que esta aplicación nos permite un diseño amplio y completo de las tablas y los campos.

Para la creación de la base de datos debe hacerse en primer lugar la creación de un usuario, luego crear una nueva BD con el nombre BD_Facturacion_Pedidos.

Script Base Datos.

```
USE [BD_Facturacion_Pedidos]
GO
/****** Object:  Table [dbo].[Tbl_Cliente]    Script Date:
10/30/2013 00:24:15 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[Tbl_Cliente](
      [Cli_Codigo] [int] NOT NULL,
      [Cli_Cedula] [varchar](50) NULL,
      [Cli_Nombre] [varchar](50) NULL,
      [Cli_Apellido] [varchar](50) NULL,
      [Cli_Direccion] [varchar](50) NULL,
      [Cli_Telefono] [varchar](50) NULL,
      [Cli_Email] [varchar](50) NULL,
      [Cli_DirFoto] [varchar](100) NULL,
      [Cli_Estado] [char](10) NULL,
 CONSTRAINT [PK_cliente] PRIMARY KEY CLUSTERED
(
```

```
	[Cli_Codigo] ASC
)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS  = ON, ALLOW_PAGE_LOCKS  = ON)
ON [PRIMARY]
) ON [PRIMARY]
GO
SET ANSI_PADDING OFF
GO
/****** Object:  Table [dbo].[Tbl_Auditoria]    Script Date:
10/30/2013 00:24:15 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[Tbl_Auditoria](
	[Aud_Codigo] [int] NOT NULL,
	[Aud_Nombre] [varchar](50) NULL,
	[Aud_FechaIngreso] [datetime] NULL,
	[Aud_FechaSalida] [datetime] NULL,
	[Aud_Usuario] [varchar](50) NULL,
	[Aud_Objeto] [varchar](50) NULL,
 CONSTRAINT [PK_auditoria65] PRIMARY KEY NONCLUSTERED
(
	[Aud_Codigo] ASC
)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS  = ON, ALLOW_PAGE_LOCKS  = ON)
ON [PRIMARY]
) ON [PRIMARY]
GO
SET ANSI_PADDING OFF
GO
/****** Object:  Table [dbo].[Tbl_Producto]    Script Date:
10/30/2013 00:24:15 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[Tbl_Producto](
	[Pro_Codigo] [int] NOT NULL,
	[Pro_Nombre] [varchar](50) NULL,
	[Pro_Descripcion] [varchar](50) NULL,
	[Pro_Precio] [decimal](10, 4) NULL,
	[Pro_Cantidad] [decimal](10, 4) NULL,
	[Pro_Stock] [decimal](10, 4) NULL,
	[Pro_Foto] [varchar](50) NULL,
	[Pro_Estado] [char](10) NULL,
 CONSTRAINT [PK_producto74] PRIMARY KEY NONCLUSTERED
(
	[Pro_Codigo] ASC
)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS  = ON, ALLOW_PAGE_LOCKS  = ON)
ON [PRIMARY]
) ON [PRIMARY]
GO
SET ANSI_PADDING OFF
GO
```

```
/****** Object:  Table [dbo].[Tbl_Tipo_Usuario]    Script Date:
10/30/2013 00:24:15 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[Tbl_Tipo_Usuario](
      [TipUsu_Codigo] [int] NOT NULL,
      [TipUsu_Nombre] [varchar](50) NULL,
      [TipUsu_Estado] [char](10) NULL,
 CONSTRAINT [PK_tipo_usuario64] PRIMARY KEY NONCLUSTERED
(
      [TipUsu_Codigo] ASC
)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS  = ON, ALLOW_PAGE_LOCKS  = ON)
ON [PRIMARY]
) ON [PRIMARY]
GO
SET ANSI_PADDING OFF
GO
/****** Object:  Table [dbo].[Tbl_Tipo_Producto]    Script Date:
10/30/2013 00:24:15 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[Tbl_Tipo_Producto](
      [TipPro_Codigo] [int] NOT NULL,
      [TipPro_Nombre] [varchar](50) NULL,
      [TipPro_Estado] [char](10) NULL,
 CONSTRAINT [PK_tipo_producto75] PRIMARY KEY NONCLUSTERED
(
      [TipPro_Codigo] ASC
)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS  = ON, ALLOW_PAGE_LOCKS  = ON)
ON [PRIMARY]
) ON [PRIMARY]
GO
SET ANSI_PADDING OFF
GO
/****** Object:  Table [dbo].[Tbl_Tipo_Pago]    Script Date:
10/30/2013 00:24:15 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[Tbl_Tipo_Pago](
      [TipPago_Codigo] [int] NOT NULL,
      [TipPago_Nombre] [varchar](50) NULL,
      [TipPago_Estado] [char](10) NULL,
 CONSTRAINT [PK_tipo_pago70] PRIMARY KEY NONCLUSTERED
(
      [TipPago_Codigo] ASC
```

```
)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS  = ON, ALLOW_PAGE_LOCKS  = ON)
ON [PRIMARY]
) ON [PRIMARY]
GO
SET ANSI_PADDING OFF
GO
/****** Object:  Table [dbo].[Tbl_Mesa]    Script Date: 10/30/2013
00:24:15 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[Tbl_Mesa](
      [Mes_Codigo] [int] NOT NULL,
      [Mes_Numero] [int] NULL,
      [Mes_Descripcion] [varchar](50) NULL,
      [Mes_Estado] [char](10) NULL,
 CONSTRAINT [PK_Tbl_Mesa] PRIMARY KEY CLUSTERED
(
      [Mes_Codigo] ASC
)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS  = ON, ALLOW_PAGE_LOCKS  = ON)
ON [PRIMARY]
) ON [PRIMARY]
GO
SET ANSI_PADDING OFF
GO
/****** Object:  Table [dbo].[Tbl_Usuario]    Script Date:
10/30/2013 00:24:15 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[Tbl_Usuario](
      [Usu_Codigo] [int] NOT NULL,
      [Usu_Cedula] [varchar](50) NULL,
      [Usu_Nombre] [varchar](50) NULL,
      [Usu_Apellido] [varchar](50) NULL,
      [Usu_Direccion] [varchar](50) NULL,
      [Usu_Telefono] [varchar](50) NULL,
      [Usu_Email] [varchar](50) NULL,
      [Usu_DirFoto] [varchar](50) NULL,
      [Usu_Estado] [char](10) NULL,
      [Usu_Usuario] [varchar](50) NULL,
      [Usu_Clave] [varchar](50) NULL,
      [TipUsu_Codigo] [int] NOT NULL,
 CONSTRAINT [PK_usuario66] PRIMARY KEY NONCLUSTERED
(
      [Usu_Codigo] ASC
)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS  = ON, ALLOW_PAGE_LOCKS  = ON)
ON [PRIMARY]
) ON [PRIMARY]
GO
SET ANSI_PADDING OFF
```

```
GO
/****** Object:  Table [dbo].[Tbl_Producto_Terminado]    Script
Date: 10/30/2013 00:24:15 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[Tbl_Producto_Terminado](
      [ProTer_Codigo] [int] NOT NULL,
      [ProTer_Nombre] [varchar](50) NULL,
      [ProTer_Descripcion] [varchar](50) NULL,
      [ProTer_Precio] [decimal](10, 2) NULL,
      [ProTer_Cantidad] [int] NULL,
      [ProTer_Stock] [int] NULL,
      [ProTer_DirFoto] [varchar](50) NULL,
      [ProTer_Estado] [char](10) NULL,
      [TipPro_Codigo] [int] NOT NULL,
 CONSTRAINT [PK_producto_terminado73] PRIMARY KEY NONCLUSTERED
(
      [ProTer_Codigo] ASC
)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS  = ON, ALLOW_PAGE_LOCKS  = ON)
ON [PRIMARY]
) ON [PRIMARY]
GO
SET ANSI_PADDING OFF
GO
/****** Object:  Table [dbo].[Tbl_Pedido]    Script Date: 10/30/2013
00:24:15 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[Tbl_Pedido](
      [Ped_Codigo] [int] NOT NULL,
      [Ped_Fecha] [datetime] NULL,
      [Ped_Estado] [char](10) NULL,
      [Ped_Estado2] [char](10) NULL,
      [Usu_Codigo] [int] NOT NULL,
      [Cli_Codigo] [int] NOT NULL,
      [Mes_Codigo] [int] NULL,
 CONSTRAINT [PK_pedido67] PRIMARY KEY NONCLUSTERED
(
      [Ped_Codigo] ASC
)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS  = ON, ALLOW_PAGE_LOCKS  = ON)
ON [PRIMARY]
) ON [PRIMARY]
GO
SET ANSI_PADDING OFF
GO
/****** Object:  Table [dbo].[Tbl_Detalle_PPTerminado]    Script
Date: 10/30/2013 00:24:15 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
```

```
GO
CREATE TABLE [dbo].[Tbl_Detalle_PPTerminado](
	[DetPPt_Codigo] [int] NOT NULL,
	[ProTer_Codigo] [int] NOT NULL,
	[Pro_Codigo] [int] NOT NULL,
 CONSTRAINT [PK_detalle_productoProdTerminado] PRIMARY KEY CLUSTERED
(
	[DetPPt_Codigo] ASC
)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS  = ON, ALLOW_PAGE_LOCKS  = ON)
ON [PRIMARY]
) ON [PRIMARY]
GO
/****** Object:  Table [dbo].[Tbl_Reservacion]    Script Date:
10/30/2013 00:24:15 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[Tbl_Reservacion](
	[Res_Codigo] [int] NOT NULL,
	[Res_Fecha] [datetime] NULL,
	[Res_NombreCliente] [varchar](50) NULL,
	[Res_Estado] [char](10) NULL,
	[Ped_Codigo] [int] NOT NULL,
 CONSTRAINT [PK_reservacion72] PRIMARY KEY NONCLUSTERED
(
	[Res_Codigo] ASC
)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS  = ON, ALLOW_PAGE_LOCKS  = ON)
ON [PRIMARY]
) ON [PRIMARY]
GO
SET ANSI_PADDING OFF
GO
/****** Object:  Table [dbo].[Tbl_Factura]    Script Date:
10/30/2013 00:24:15 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[Tbl_Factura](
	[Fac_Codigo] [int] NOT NULL,
	[Fac_Fecha] [datetime] NULL,
	[Fac_FechaVencimiento] [datetime] NULL,
	[Fac_Impuesto] [decimal](10, 4) NULL,
	[Fac_Estado] [char](10) NULL,
	[Ped_Codigo] [int] NOT NULL,
	[TipPago_Codigo] [int] NOT NULL,
	[Fac_Total] [decimal](10, 4) NULL,
 CONSTRAINT [PK_factura69] PRIMARY KEY NONCLUSTERED
(
	[Fac_Codigo] ASC
)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS  = ON, ALLOW_PAGE_LOCKS  = ON)
ON [PRIMARY]
```

```
) ON [PRIMARY]
GO
SET ANSI_PADDING OFF
GO
/****** Object:  Table [dbo].[Tbl_Detalle_Pedido]    Script Date:
10/30/2013 00:24:15 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[Tbl_Detalle_Pedido](
      [DetPed_Codigo] [int] NOT NULL,
      [DetPed_Cantidad] [int] NULL,
      [DetPed_Precio] [decimal](18, 2) NULL,
      [DetPed_Descuento] [decimal](18, 2) NULL,
      [DetPed_Fecha] [datetime] NULL,
      [DetPed_Estado] [char](10) NULL,
      [ProTer_Codigo] [int] NOT NULL,
      [Ped_Codigo] [int] NOT NULL,
 CONSTRAINT [PK_detalle_pedido71] PRIMARY KEY NONCLUSTERED
(
      [DetPed_Codigo] ASC
)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS  = ON, ALLOW_PAGE_LOCKS  = ON)
ON [PRIMARY]
) ON [PRIMARY]
GO
SET ANSI_PADDING OFF
GO
/****** Object:  ForeignKey [FK_usuario48]    Script Date:
10/30/2013 00:24:15 ******/
ALTER TABLE [dbo].[Tbl_Usuario]  WITH CHECK ADD  CONSTRAINT
[FK_usuario48] FOREIGN KEY([TipUsu_Codigo])
REFERENCES [dbo].[Tbl_Tipo_Usuario] ([TipUsu_Codigo])
GO
ALTER TABLE [dbo].[Tbl_Usuario] CHECK CONSTRAINT [FK_usuario48]
GO
/****** Object:  ForeignKey [FK_producto_terminado57]    Script
Date: 10/30/2013 00:24:15 ******/
ALTER TABLE [dbo].[Tbl_Producto_Terminado]  WITH CHECK ADD
CONSTRAINT [FK_producto_terminado57] FOREIGN KEY([TipPro_Codigo])
REFERENCES [dbo].[Tbl_Tipo_Producto] ([TipPro_Codigo])
GO
ALTER TABLE [dbo].[Tbl_Producto_Terminado] CHECK CONSTRAINT
[FK_producto_terminado57]
GO
/****** Object:  ForeignKey [FK_pedido_usuario]    Script Date:
10/30/2013 00:24:15 ******/
ALTER TABLE [dbo].[Tbl_Pedido]  WITH CHECK ADD  CONSTRAINT
[FK_pedido_usuario] FOREIGN KEY([Usu_Codigo])
REFERENCES [dbo].[Tbl_Usuario] ([Usu_Codigo])
GO
ALTER TABLE [dbo].[Tbl_Pedido] CHECK CONSTRAINT [FK_pedido_usuario]
GO
/****** Object:  ForeignKey [FK_pedido49]    Script Date: 10/30/2013
00:24:15 ******/
ALTER TABLE [dbo].[Tbl_Pedido]  WITH CHECK ADD  CONSTRAINT
[FK_pedido49] FOREIGN KEY([Cli_Codigo])
```

```
REFERENCES [dbo].[Tbl_Cliente] ([Cli_Codigo])
GO
ALTER TABLE [dbo].[Tbl_Pedido] CHECK CONSTRAINT [FK_pedido49]
GO
/****** Object:  ForeignKey [FK_Tbl_Pedido_Tbl_Mesa]    Script Date:
10/30/2013 00:24:15 ******/
ALTER TABLE [dbo].[Tbl_Pedido]  WITH CHECK ADD  CONSTRAINT
[FK_Tbl_Pedido_Tbl_Mesa] FOREIGN KEY([Mes_Codigo])
REFERENCES [dbo].[Tbl_Mesa] ([Mes_Codigo])
GO
ALTER TABLE [dbo].[Tbl_Pedido] CHECK CONSTRAINT
[FK_Tbl_Pedido_Tbl_Mesa]
GO
/****** Object:  ForeignKey
[FK_detalle_productoProdTerminado_producto]    Script Date:
10/30/2013 00:24:15 ******/
ALTER TABLE [dbo].[Tbl_Detalle_PPTerminado]  WITH CHECK ADD
CONSTRAINT [FK_detalle_productoProdTerminado_producto] FOREIGN
KEY([Pro_Codigo])
REFERENCES [dbo].[Tbl_Producto] ([Pro_Codigo])
GO
ALTER TABLE [dbo].[Tbl_Detalle_PPTerminado] CHECK CONSTRAINT
[FK_detalle_productoProdTerminado_producto]
GO
/****** Object:  ForeignKey
[FK_detalle_productoProdTerminado_producto_terminado]    Script
Date: 10/30/2013 00:24:15 ******/
ALTER TABLE [dbo].[Tbl_Detalle_PPTerminado]  WITH CHECK ADD
CONSTRAINT [FK_detalle_productoProdTerminado_producto_terminado]
FOREIGN KEY([ProTer_Codigo])
REFERENCES [dbo].[Tbl_Producto_Terminado] ([ProTer_Codigo])
GO
ALTER TABLE [dbo].[Tbl_Detalle_PPTerminado] CHECK CONSTRAINT
[FK_detalle_productoProdTerminado_producto_terminado]
GO
/****** Object:  ForeignKey [FK_reservacion_pedido]    Script Date:
10/30/2013 00:24:15 ******/
ALTER TABLE [dbo].[Tbl_Reservacion]  WITH CHECK ADD  CONSTRAINT
[FK_reservacion_pedido] FOREIGN KEY([Ped_Codigo])
REFERENCES [dbo].[Tbl_Pedido] ([Ped_Codigo])
GO
ALTER TABLE [dbo].[Tbl_Reservacion] CHECK CONSTRAINT
[FK_reservacion_pedido]
GO
/****** Object:  ForeignKey [FK_factura51]    Script Date:
10/30/2013 00:24:15 ******/
ALTER TABLE [dbo].[Tbl_Factura]  WITH CHECK ADD  CONSTRAINT
[FK_factura51] FOREIGN KEY([Ped_Codigo])
REFERENCES [dbo].[Tbl_Pedido] ([Ped_Codigo])
GO
ALTER TABLE [dbo].[Tbl_Factura] CHECK CONSTRAINT [FK_factura51]
GO
/****** Object:  ForeignKey [FK_factura53]    Script Date:
10/30/2013 00:24:15 ******/
ALTER TABLE [dbo].[Tbl_Factura]  WITH CHECK ADD  CONSTRAINT
[FK_factura53] FOREIGN KEY([TipPago_Codigo])
REFERENCES [dbo].[Tbl_Tipo_Pago] ([TipPago_Codigo])
GO
ALTER TABLE [dbo].[Tbl_Factura] CHECK CONSTRAINT [FK_factura53]
GO
```

```
/****** Object:  ForeignKey [FK_detalle_pedido50]    Script Date:
10/30/2013 00:24:15 ******/
ALTER TABLE [dbo].[Tbl_Detalle_Pedido]  WITH CHECK ADD  CONSTRAINT
[FK_detalle_pedido50] FOREIGN KEY([Ped_Codigo])
REFERENCES [dbo].[Tbl_Pedido] ([Ped_Codigo])
GO
ALTER TABLE [dbo].[Tbl_Detalle_Pedido] CHECK CONSTRAINT
[FK_detalle_pedido50]
GO
/****** Object:  ForeignKey [FK_detalle_pedido56]    Script Date:
10/30/2013 00:24:15 ******/
ALTER TABLE [dbo].[Tbl_Detalle_Pedido]  WITH CHECK ADD  CONSTRAINT
[FK_detalle_pedido56] FOREIGN KEY([ProTer_Codigo])
REFERENCES [dbo].[Tbl_Producto_Terminado] ([ProTer_Codigo])
GO
ALTER TABLE [dbo].[Tbl_Detalle_Pedido] CHECK CONSTRAINT
[FK_detalle_pedido56]
GO
```

3. Desarrollo del Web Services

El diseño del servicio web se realizó en Visual Studio 2010, el mismo que está realizado en 3 capas, las cuales se indican a continuación.

Para realizar la conexión a la base de datos debe crearse el origen de datos y luego cargarse el dbml.

Codificación Servicio Web.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Services;
using System.Xml.Linq;
using System.Web.Script.Services;
using System.Xml.Serialization;
using System.Data;
using System.Web.Script.Serialization;
using System.Web.Services.Protocols;
using LogicaNegocios;
using Modelos;

namespace WS_FacturacionPedidos
{
    /// <summary>
    /// Descripción breve de Clientes
    /// </summary>
    [WebService(Namespace = "elhaker.com.ec")]
    [WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
    [System.ComponentModel.ToolboxItem(false)]
    // Para permitir que se llame a este servicio Web desde un script, usando
ASP.NET AJAX, quite la marca de comentario de la línea siguiente.
    // [System.Web.Script.Services.ScriptService]

    public class Clientes : System.Web.Services.WebService
    {
        //Metodo del web services Tipo de Usuario lista de tipos de
usuarios.....................
        [WebMethod(Description = "Provee lista Clientes")]
        [ScriptMethod(ResponseFormat = ResponseFormat.Json)]
        public string ListarClientes()  //Presenta lista de clientes
        {
            Logica_Cliente InfoCliente = new Logica_Cliente();//llama a la
logica y presenta datos
             return InfoCliente.ListarClientesInfo();
        }
//.....................................................................
        //Metodo del web services que devuelve una lista
simple.....................
        [WebMethod(Description = "Provee lista Simple de Clientes")]
        [ScriptMethod(ResponseFormat = ResponseFormat.Json)]

        public string ListaSimpleClientes()  //Presenta lista de clientes
        {
            Logica_Cliente InfoCliente = new Logica_Cliente();//llama a la
logica y presenta datos
            return InfoCliente.ListaSimpleClientesInfo();
```

```
        }
//.....................................................................
        //Metodo para buscar cliente X ID

//*************************************************************************
        [WebMethod(Description = "Busca Clientes X ID")]
        [ScriptMethod(ResponseFormat = ResponseFormat.Json)]

        public string BuscarClientesXId(String codigo)
        {
            Logica_Cliente InfoCliente = new Logica_Cliente();//llama a la
logica y presenta datos
            return(InfoCliente.BuscarClientesXidInfo( codigo));
        }
//*************************************************************************
        //Metodo para buscar cliente X ID completo
//*************************************************************************
        [WebMethod(Description = "Busca Clientes X ID completo")]
        [ScriptMethod(ResponseFormat = ResponseFormat.Json)]

        public string BuscarClientesXIdCompleto(String codigo)
        {
            Logica_Cliente InfoCliente = new Logica_Cliente();//llama a la
logica y presenta datos
            return (InfoCliente.BuscarClientesXIdCompleto(codigo));
        }
//*************************************************************************
        //Metodo para registrar un nuevo cliente
//*************************************************************************
        [WebMethod(Description = "Registrar un nuevo cliente")]
        [ScriptMethod(ResponseFormat = ResponseFormat.Json)]
        public string RegistrarClientes(String Cedula, string Nombre,string
Apellido,string Direccion,
            string Telefono,string Email,string DirFoto)
        {
            Logica_Cliente InfoCliente = new Logica_Cliente();//llama a la
logica e inserta datos
            return (InfoCliente.InsertarCliente(
Cedula,Nombre,Apellido,Direccion,
            Telefono,Email,DirFoto));
        }
//*************************************************************************
        //Metodo para actualizar un cliente
//*************************************************************************
        [WebMethod(Description = "Actualizar datos del cliente especificado
por su ID")]
        [ScriptMethod(ResponseFormat = ResponseFormat.Json)]
        public string ActualizarClientes(int Codigo, String Cedula, string
Nombre, string Apellido, string Direccion,
            string Telefono, string Email, string DirFoto)
        {
            Logica_Cliente InfoCliente = new Logica_Cliente();//llama a la
logica e inserta datos
            return (InfoCliente.ActualizarCliente(Codigo,Cedula, Nombre,
Apellido, Direccion,
            Telefono, Email, DirFoto));
        }
//*************************************************************************
        //Metodo para eliminar datos
//*************************************************************************
        [WebMethod(Description = "Eliminar datos de clientes")]
```

```
        [ScriptMethod(ResponseFormat = ResponseFormat.Json)]

        public string EliminarClientes(int Codigo)
        {
            Logica_Cliente InfoCliente = new Logica_Cliente();
            return (InfoCliente.EliminarClienteInfo(Codigo));//envia codigo
para eliminar y devuelme mensaje
        }
//*************************************************************************
        //Metodo para buscar cliente X apellido
//*************************************************************************
        [WebMethod(Description = "Busca Clientes X Apellido")]
        [ScriptMethod(ResponseFormat = ResponseFormat.Json)]

        public string BuscarClientesXApellido(String Apellido)
        {
            Logica_Cliente InfoCliente = new Logica_Cliente();//llama a la
logica y presenta datos
            return (InfoCliente.BuscarClientesXApellido(Apellido));
        }
//*************************************************************************
***
    }
}
```

Lógica del Servicio Web.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Modelos;
using AccesoDatos;
using System.Xml.Linq;
using System.Web.Script.Services;
using System.Xml.Serialization;
using System.Data;
using System.Web.Script.Serialization;
using System.Web.Services.Protocols;
using System.Web;
namespace LogicaNegocios
{
    public class Logica_Cliente
    {
        DBAccesoDatosDataContext BD = new DBAccesoDatosDataContext();//llama
al acceso a datos
        //Extrae lista de clientes
        //*************************************************************************
        public  string ListarClientesInfo()
        {
            List<ListarClientes> Lista = new List<ListarClientes>(); //llama a
modelos por id o nombre
            foreach (Tbl_Cliente item in BD.Tbl_Cliente.ToList())
            {
                ListarClientes NuevaLista = new ListarClientes();
                NuevaLista.InfoCodigo = item.Cli_Codigo;
                NuevaLista.InfoCedula = item.Cli_Cedula;
                NuevaLista.InfoNombre = item.Cli_Nombre;
                NuevaLista.InfoApellido = item.Cli_Apellido;
                NuevaLista.InfoDireccion = item.Cli_Direccion;
                NuevaLista.InfoTelefono = item.Cli_Telefono;
```

```
                NuevaLista.InfoEmail = item.Cli_Telefono;
                NuevaLista.InfoDirFoto = item.Cli_DirFoto;
               // NuevaLista.InfoEstado = item.Cli_Estado;
                Lista.Add(NuevaLista);
            }
            //Guarda en un archivo de texto los datos logicos de quien
solicito lista clientes
            //*********************
            RegSolicitudes.Reg_Clientes("El cliente: " +
HttpContext.Current.Request.UserAgent +
                " (" + HttpContext.Current.Request.UserHostName +
                ") solicito toda la lista de clientes." );
            return new JavaScriptSerializer().Serialize(Lista); //presenta
datos en xml
           //***********************
        }
//*********************************************************************
        //Extrae lista simple de clientes
        //*********************************************************************
        public string ListaSimpleClientesInfo()
        {
            List<ListaSimple> Lista = new List<ListaSimple>(); //llama a
modelos por id o nombre
            foreach (Tbl_Cliente item in BD.Tbl_Cliente.ToList())
            {
                ListaSimple NuevaLista = new ListaSimple();
                NuevaLista.InfoCodigo = item.Cli_Codigo;
                NuevaLista.InfoNombre = item.Cli_Nombre;
                NuevaLista.InfoApellido = item.Cli_Apellido;

                Lista.Add(NuevaLista);
            }
            //Guarda en un archivo de texto los datos logicos de quien
solicito lista clientes
            //*********************
            RegSolicitudes.Reg_Clientes("El cliente: " +
HttpContext.Current.Request.UserAgent +
                " (" + HttpContext.Current.Request.UserHostName +
                ") solicito una lista simple de clientes.");
            return new JavaScriptSerializer().Serialize(Lista); //presenta
datos en xml
            //***********************
        }
//*********************************************************************
        //Busca clientes por ID
//*********************************************************************
        public string BuscarClientesXidInfo(string Codigo)
        {
            try
            {
                var Buscar = BD.Tbl_Cliente.Where(c =>
c.Cli_Codigo.Equals(Codigo)).FirstOrDefault();
                if (Buscar != null)
                {
                    Lista_IdNombre AuxBuscar = new Lista_IdNombre();
                    AuxBuscar.InfoCodigo = Buscar.Cli_Codigo;
                    AuxBuscar.InfoNombre = Buscar.Cli_Nombre;
                    //Guarda en un archivo de texto los datos logicos de quien
busco lista clientes
                    //*********************
```

```
                    RegSolicitudes.Reg_Clientes("El cliente: " +
HttpContext.Current.Request.UserAgent +
                        " (" + HttpContext.Current.Request.UserHostName +
                        ") solicito toda la lista de clientes.");
                    //**************************
                    return new JavaScriptSerializer().Serialize(AuxBuscar);
                }
                else
                    return "{\"error\": true, \"encontrado\":false}";
            }
            catch
            {
                return "Error al Buscar";
            }
        }

//*********************************************************************
        //Busca clientes por ID con todos datos
//*********************************************************************
        public string BuscarClientesXIdCompleto(string Codigo)
        {
            try
            {
                var item = BD.Tbl_Cliente.Where(c =>
c.Cli_Codigo.Equals(Codigo)).FirstOrDefault();
                if (item != null)
                {
                ListarClientes NuevaLista = new ListarClientes();
                NuevaLista.InfoCodigo = item.Cli_Codigo;
                NuevaLista.InfoCedula = item.Cli_Cedula;
                NuevaLista.InfoNombre = item.Cli_Nombre;
                NuevaLista.InfoApellido = item.Cli_Apellido;
                NuevaLista.InfoDireccion = item.Cli_Direccion;
                NuevaLista.InfoTelefono = item.Cli_Telefono;
                NuevaLista.InfoEmail = item.Cli_Telefono;
                NuevaLista.InfoDirFoto = item.Cli_DirFoto;
               // NuevaLista.InfoEstado = item.Cli_Estado;
                    //Guarda en un archivo de texto los datos logicos de quien
busco lista clientes
                    //********************
                    RegSolicitudes.Reg_Clientes("El cliente: " +
HttpContext.Current.Request.UserAgent +
                        " (" + HttpContext.Current.Request.UserHostName +
                        ") busco una lista de clientes.");
                    //**************************
                    return new JavaScriptSerializer().Serialize(NuevaLista);
                }
                else
                    return "{\"error\": true, \"encontrado\":false}";
            }
            catch
            {
                return "Error al Buscar";
            }
        }
//*********************************************************************
        //Registrar un nuevo cliente
//*********************************************************************
        public string InsertarCliente(String Cedula, string Nombre, string
Apellido, string Direccion,
            string Telefono, string Email, string DirFoto)
```

```
        {
            ResultadoInsertar resultadoInsert = new
ResultadoInsertar();//devuelve mensaje de insercion
            try
            {
                Tbl_Cliente cliente = new Tbl_Cliente();//crear el objeto
cliente
                BD.ExecuteCommand("Insert into
Tbl_Cliente(Cli_Codigo,Cli_Cedula,Cli_Nombre,Cli_Apellido,"+
                "Cli_Direccion,Cli_Telefono,Cli_Email,Cli_DirFoto,Cli_Estado)"
+   //guardar un nuevo cliente a bd
                    "values({0},{1},{2},{3},{4},{5},{6},{7},{8})", new
object[]
                    {

cliente.Cli_Codigo=Incrementa_Codigo.Codigo_Clientes(),
                        cliente.Cli_Cedula = Cedula,
                        cliente.Cli_Nombre=Nombre,
                        cliente.Cli_Apellido=Apellido,
                        cliente.Cli_Direccion=Direccion,
                        cliente.Cli_Telefono=Telefono,
                        cliente.Cli_Email=Email,
                        cliente.Cli_DirFoto=DirFoto,
                        cliente.Cli_Estado="A"
                    });
                resultadoInsert.InfoError = false; //retornar mensaje correcto
o de error y el codigo
                resultadoInsert.InfoRealizado = true;
                resultadoInsert.InfoIDRegistro = cliente.Cli_Codigo; //toma
codigo de insercion
                resultadoInsert.InfoMensaje = "Cliente registrado
correctamente."; //mensaje
                //Guarda en un archivo de texto los datos logicos de quien
registro un nuevo clientes
                //*******************************
                RegSolicitudes.Reg_Clientes("El cliente: " +
HttpContext.Current.Request.UserAgent +
                                            " (" +
HttpContext.Current.Request.UserHostName +
                                               ") registro un nuevo cliente >>" +
cliente.Cli_Codigo);
                //*******************************
                return new JavaScriptSerializer().Serialize(resultadoInsert);
//retorna valores en xml
            }
            catch (Exception) //error si falla al insertar registro
            {
                resultadoInsert.InfoError = true;
                resultadoInsert.InfoRealizado = false;
                resultadoInsert.InfoIDRegistro = 0; //codigo de insercion 0 no
se realizo
                resultadoInsert.InfoMensaje = "Error al Registrar el Nuevo
Cliente."; //mensaje
                return new JavaScriptSerializer().Serialize(resultadoInsert);
//retorna valores en xml
            }
        }
//**************************************************************************
*****************
```

```
        //Actualizar un cliente
//*************************************************************************
***+****
        public string ActualizarCliente(int Codigo ,String Cedula, string
Nombre, string Apellido, string Direccion,
            string Telefono, string Email, string DirFoto)
        {
            ResultadoInsertar ResultadoServicio = new ResultadoInsertar();
            //Recuperar el objeto cliente si existe, realiza los cambios y
guarda
            Tbl_Cliente cliente = new Tbl_Cliente();
            try
            {
                BD.ExecuteCommand("update Tbl_Cliente set
Cli_Cedula={0},Cli_Nombre={1},Cli_Apellido={2}," +

"Cli_Direccion={3},Cli_Telefono={4},Cli_Email={5},Cli_DirFoto={6} where
Cli_Codigo={7}", new object[]
                 {
                          cliente.Cli_Cedula = Cedula,
                          cliente.Cli_Nombre=Nombre,
                          cliente.Cli_Apellido=Apellido,
                          cliente.Cli_Direccion=Direccion,
                          cliente.Cli_Telefono=Telefono,
                          cliente.Cli_Email=Email,
                          cliente.Cli_DirFoto=DirFoto,
                          cliente.Cli_Codigo=Codigo
                 });
                ResultadoServicio.InfoError = false;
                ResultadoServicio.InfoRealizado = true;
                ResultadoServicio.InfoIDRegistro = Codigo;
                ResultadoServicio.InfoMensaje = "Cambios realizados
correctamente";
                //Guarda en un archivo de texto los datos logicos de quien
pidio modificar clientes
                //*********************
                RegSolicitudes.Reg_Clientes("El cliente: " +
HttpContext.Current.Request.UserAgent +
                    " (" + HttpContext.Current.Request.UserHostName + ")
modificó los datos del cliente: " +
                                                                    Codigo);
                //**************************
                return new
JavaScriptSerializer().Serialize(ResultadoServicio);
            }
            catch
            {
                ResultadoServicio.InfoError = true;
                ResultadoServicio.InfoRealizado = false;
                ResultadoServicio.InfoIDRegistro = Codigo;
                ResultadoServicio.InfoMensaje = "No se encontró el registro.";
                return new
JavaScriptSerializer().Serialize(ResultadoServicio);
            }
        }
//*************************************************************************
        //Elimina registros
//*************************************************************************
        public string EliminarClienteInfo(int Codigo)
        {
            ResultadoInsertar ResultadoServicio = new ResultadoInsertar();
```

```
            Tbl_Cliente auxElimina = new Tbl_Cliente();
            try
            {
                BD.ExecuteCommand("update Tbl_Cliente set Cli_Estado={0} where
Cli_Codigo={1}", new object[] //ejecuta eliminar
                 {
                          auxElimina.Cli_Estado = "E",
                          auxElimina.Cli_Codigo=Codigo
                 });
                ResultadoServicio.InfoError = false;
                ResultadoServicio.InfoRealizado = true;
                ResultadoServicio.InfoIDRegistro = Codigo;
                ResultadoServicio.InfoMensaje = "Eliminado correctamente";

                RegSolicitudes.Reg_Clientes("El cliente: " +
HttpContext.Current.Request.UserAgent +
                    " (" + HttpContext.Current.Request.UserHostName + ")
elimino  el registro: " +
                                                                    Codigo);
//guarda en txt datos solicitud
                return new
JavaScriptSerializer().Serialize(ResultadoServicio);//devuelve datos en
formato.json

            }
            catch
            {
                ResultadoServicio.InfoError = true;
                ResultadoServicio.InfoRealizado = false;
                ResultadoServicio.InfoIDRegistro = Codigo;
                ResultadoServicio.InfoMensaje = "No se elimino el registro.";
                return new
JavaScriptSerializer().Serialize(ResultadoServicio);
            }
        }
//*************************************************************************
        //Busca clientes por apellido
//*************************************************************************
        public string BuscarClientesXApellido(string Apellido)
        {
            try
            {
                List<ListaSimple> Lista = new List<ListaSimple>(); //llama a
modelos por id o nombre
                foreach (Tbl_Cliente item in BD.Tbl_Cliente.Where(c =>
c.Cli_Apellido.Equals(Apellido)).ToList())
                {
                    ListaSimple NuevaLista = new ListaSimple();
                    NuevaLista.InfoCodigo = item.Cli_Codigo;
                    NuevaLista.InfoNombre = item.Cli_Nombre;
                    NuevaLista.InfoApellido = item.Cli_Apellido;
                    Lista.Add(NuevaLista);
                }
                //Guarda en un archivo de texto los datos logicos de quien
busco lista clientes
                //*********************
                RegSolicitudes.Reg_Clientes("El cliente: " +
HttpContext.Current.Request.UserAgent +
                    " (" + HttpContext.Current.Request.UserHostName +
                    ") solicito una busqueda por apellido.");
                //**************************
```

```
                return new JavaScriptSerializer().Serialize(Lista);
            }
            catch
            {
                return "Error al Buscar";
            }
        }
//*********************************************************************
    }
}
```

Modelos de Servicio Web.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using AccesoDatos;
namespace Modelos
{
    public class ListarClientes
    {
        //--------Debuelve y pide datos de los clientes
            //pide y devuelve codigo.................
            private int Codigo;
            public int InfoCodigo
            {
                get { return Codigo; }
                set { Codigo = value; }
            }
            //pide y devuelve cedula.................
            private string Cedula;
            public string InfoCedula
            {
                get { return Cedula; }
                set { Cedula = value; }
            }
            //pide y devuelve nombre.....................
            private string Nombre;
            public string InfoNombre
            {
                get { return Nombre; }
                set { Nombre = value; }
            }
            //pide y devuelve apellido.....................
            private string Apellido;
            public string InfoApellido
            {
                get { return Apellido; }
                set { Apellido = value; }
            }
            //pide y devuelve direccion.....................
            private string Direccion;
            public string InfoDireccion
            {
                get { return Direccion; }
                set { Direccion = value; }
            }
            //pide y devuelve Telefono.....................
            private string Telefono;
            public string InfoTelefono
```

```
            {
                get { return Telefono; }
                set { Telefono = value; }
            }
            //pide y devuelve Email.....................
            private string Email;
            public string InfoEmail
            {
                get { return Email; }
                set { Email = value; }
            }
            //pide y devuelve DirFoto.....................
            private string DirFoto;
            public string InfoDirFoto
            {
                get { return DirFoto; }
                set { DirFoto = value; }
            }
            //pide y devuelve Estado.....................
            private string Estado;
            public string InfoEstado
            {
                get { return Estado; }
                set { Estado = value; }
            }
        }
        // ------------------------------------------------------------------
    }
```

4. Desarrollo de la aplicación

La aplicación fue desarrollada en Android 4.0, la misma que se encuentra distribuida en varios paquetes, los cuales se indican a continuación.

Para realzarse la conexión al servicio web, debe cambiarse la ip, el usuario y la clave de inicio de sesión del servidor, en la clase ConexionServicioWeb.java

Codificación Layouts.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical" >
        <include
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            layout="@layout/cab_listado" />
    </LinearLayout>
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
```

```
        android:layout_weight="0.35"
        android:background="@drawable/fondo_layout"
        android:orientation="vertical" >
        <TextView
            android:id="@+id/textView1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="BUSCAR APELLIDO"
            android:textAppearance="?android:attr/textAppearanceLarge" />
        <EditText
            android:id="@+id/BusCliTxtNombre"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:background="@drawable/estilo_textos"
            android:ems="10" >
        </EditText>
        <ImageButton
            android:id="@+id/ImgBtnListar"
            android:layout_width="match_parent"
            android:layout_height="50dp"
            android:layout_gravity="center"
            android:background="@drawable/estilo_botones"
            android:src="@android:drawable/ic_menu_view" />
        <ListView
            android:id="@+id/ListCli"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:textFilterEnabled="true" >
        </ListView>
    </LinearLayout>
</LinearLayout>
```

Codificación Claces Java.

```
package com.android.appfacturacionpedidos.actividades;
import java.util.HashMap;
import java.util.Map;
import android.app.Activity;
import android.app.ProgressDialog;
import android.content.Context;
import android.content.Intent;
import android.os.AsyncTask;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.ImageButton;
import android.widget.ListView;
import android.widget.TextView;
import android.widget.Toast;
import com.android.appfacturacionpedidos.R;
import com.android.appfacturacionpedidos.accesodatos.AccesoServicioWeb;
import com.android.appfacturacionpedidos.adaptadores.AdaptadorCliente;
import com.android.appfacturacionpedidos.modelos.ListaSimple;
```

```
import com.google.gson.Gson;

public class ClientesActivity extends Activity {
              ImageButton CliImgBtnAgregar;
              ImageButton CliImgBtnAtras;
              ImageButton ImgBtnListar;
       ListView listaSimpleCli;
       TextView txtbuscar;
       Integer tipoAccion;
       ListaSimple[] listSimple;
       protected static final int REQUEST_NEW_CLI = 1;
       protected static final int REQUEST_CLIENTE = 2;
       Integer UsuCod;
       AccesoServicioWeb accesoWeb=new AccesoServicioWeb();
       Gson gson;
       private Context context;
       private String res;
       private ProgressDialog pd
       @Override
       protected void onCreate(Bundle savedInstanceState) {
              super.onCreate(savedInstanceState);
              context=this;
             setContentView(R.layout.clientes);
             Bundle extras = getIntent().getExtras();
                tipoAccion = extras.getInt("tipoAccion");
             txtbuscar=(TextView)findViewById(R.id.BusCliTxtNombre);
             listaSimpleCli=(ListView)findViewById(R.id.ListCli);
             new donwloadTask1().execute("");
             pd=ProgressDialog.show(context, "Por favor espere","Consultando
clientes", true,false);
//*********************************************************
              //boton agregar
             //*********************************************************

CliImgBtnAgregar=(ImageButton)findViewById(R.id.CabLisBtnAgregar);
              CliImgBtnAgregar.setOnClickListener(new OnClickListener() {
                           @Override
                     public void onClick(View v) {
                           Intent newcliente=new
Intent(ClientesActivity.this, NuevoClienteActivity.class);
                           int texto=0;
                           newcliente.putExtra("dato",texto);
                           newcliente.putExtra("tipoAccion",0);
                           startActivity(newcliente);
                           finish();
                     }
              });
              //*********************************************************
              //boton atras
             //*********************************************************
        CliImgBtnAtras=(ImageButton)findViewById(R.id.CabLisBtnAtras);
              CliImgBtnAtras.setOnClickListener(new OnClickListener()
                                 @Override
                     public void onClick(View v) {
                           new donwloadTask1().execute("");
                           pd=ProgressDialog.show(context, "Por favor
espere","Consultando clientes", true,false);
                     }
```

```
            });
//*********************************************************
             //boton ver lista
            //*********************************************************
             ImgBtnListar=(ImageButton)findViewById(R.id.ImgBtnListar);
           ImgBtnListar.setOnClickListener(new OnClickListener() {
                  @Override
                  public void onClick(View v) {
                        new donwloadTask().execute("");
                        pd=ProgressDialog.show(context, "Por favor
espere","Consultando clientes", true,false);
                  }
            });
      }
      @Override
         protected void onActivityResult(int requestCode, int resultCode,
Intent data) {
            super.onActivityResult(requestCode, resultCode, data);
            if (requestCode == REQUEST_NEW_CLI) {
                    String CodCli=data.getStringExtra("CodCli");
                    String NombreCli=data.getStringExtra("NombreCli");

                    Intent intent = new Intent();
                      intent.putExtra("CodCli",CodCli);
                      intent.putExtra("NombreCli",NombreCli);
                      setResult(REQUEST_CLIENTE, intent);
                      finish();
                  }
         }
      //*********************************************************
      //metodo crear lista
      //*********************************************************
            private void crearLista(String strJson){
                  try{
                        if(!strJson.equals(""))
                        {
                              pd.dismiss();

                              Gson gson = new Gson();
                              listSimple=gson.fromJson(strJson,
ListaSimple[].class);
                        }
                        if(listSimple != null)
                       {
                              this.listaSimpleCli.setAdapter(new
AdaptadorCliente(this, listSimple));
                              }else

                       Toast.makeText(this, "La lista de clientes esta
vacia, o no se pudo cargar la lista de clientes (verifique que la URL de
los WS sea correcta).", Toast.LENGTH_SHORT).show();
                        }catch(Exception e)
                        {
                       Log.d("Error en ClientesActivity", e.getMessage()
+ " >>>" + e.getStackTrace());
                              Toast.makeText(this, "Ocurrio un error al
cargar la lista de clientes.", Toast.LENGTH_SHORT).show();
```

```
                        }
                };
                //metodo donwloadTask

        //*********************************************************
                        private class donwloadTask extends
AsyncTask<String, Void, Object>
                        {
                                protected String doInBackground(String... args)
{
                                        try{

                                        //se invoca nuestro metodo
                                        Map<String, String> parametros = new
HashMap<String, String>();

        parametros.put("Apellido",txtbuscar.getText().toString());
                                        Log.d("Resultadoxx: ",""+
txtbuscar.getText().toString());
                                        String
resultadoConeccion=accesoWeb.Coneccion(getString(R.string.Namespace),getSt
ring(R.string.MetodoBuscarClientes),getString(R.string.Url)+getString(R.st
ring.WsClientes),parametros );
                                        res=resultadoConeccion;
                                        Log.d("Resultadoxx: ", res);
                                        //crearLista(resultadoConeccion);
                                        return resultadoConeccion;          }
                                        catch(Exception e)
                                        {
                                                return "Error";
                                        }
                                }
                                protected void onPostExecute(Object result) {
                                        pd.dismiss();
                                        crearLista(res);
                                        //se muestra mensaje de respuesta del
servico web
                                        Toast.makeText(context, "Clientes Cargados
Correctamente", Toast.LENGTH_LONG).show();
                                        super.onPostExecute(result);
                                }

        }//*********************************************************

//*********************************************************
                //metodo donwloadTask
                //*********************************************************
                private class donwloadTask1 extends AsyncTask<String, Void,
Object>
        {
                protected String doInBackground(String... args) {
                        try{
                        //se invoca nuestro metodo

                        String
resultadoConeccion=accesoWeb.Coneccion(getString(R.string.Namespace),getSt
ring(R.string.MetodoListaSimple),getString(R.string.Url)+getString(R.strin
g.WsClientes),null );
```

```
                    res=resultadoConeccion;
                    //crearLista(resultadoConeccion);
                    return resultadoConeccion;
                          }
                    catch(Exception e)
                    {
                          return "Error";
                    }
              }
              protected void onPostExecute(Object result) {
                    pd.dismiss();
                    crearLista(res);
                    //se muestra mensaje de respuesta del servico web
                    Toast.makeText(context, "Clientes Cargados
Correctamente", Toast.LENGTH_LONG).show();
                    super.onPostExecute(result);

              //seleccion de la lista
              //*********************************************************
                    listaSimpleCli.setOnItemClickListener(new
OnItemClickListener() {
                                  @Override
                                  public void onItemClick(AdapterView<?>
pariente, View View,int posicion, long id) {
                                    ListaSimple elegido = (ListaSimple)
pariente.getItemAtPosition(posicion);
                                    int texto = elegido.getCodigo();
                                    Log.d("Parametropasa: ",""+ texto);
                                    if(tipoAccion==1){
                                   Intent inten = new
Intent(ClientesActivity.this, NuevoClienteActivity.class);
                             inten.putExtra("dato",texto);
                             inten.putExtra("tipoAccion",1);
                             startActivityForResult(inten, REQUEST_NEW_CLI);
                                    }
                                    else{
                                         Intent inten = new
Intent(ClientesActivity.this, NuevoClienteActivity.class);
                                          inten.putExtra("dato",texto);
                                          inten.putExtra("tipoAccion",0);
                                        startActivity(inten);
                                        finish();
                                    }
                        }
                              });
              }
        }
}
```

A.010. Manual de Usuario

1.01. Introducción

Este documento pretende instruir a los usuarios sobre cómo funciona la herramienta a grandes rasgos. Con esto queremos que todo el mundo conozca el funcionamiento así como la funcionalidad que proporciona la aplicación.

1.02. Ingreso a la aplicación

Figura 29. Para tener acceso, el usuario debe ingresar su usuario, contraseña y tocar el botón Ingresar, luego de 3 intentos erróneos de inicio de sesión se cerrara la aplicación.

1.03. Menú de usuario

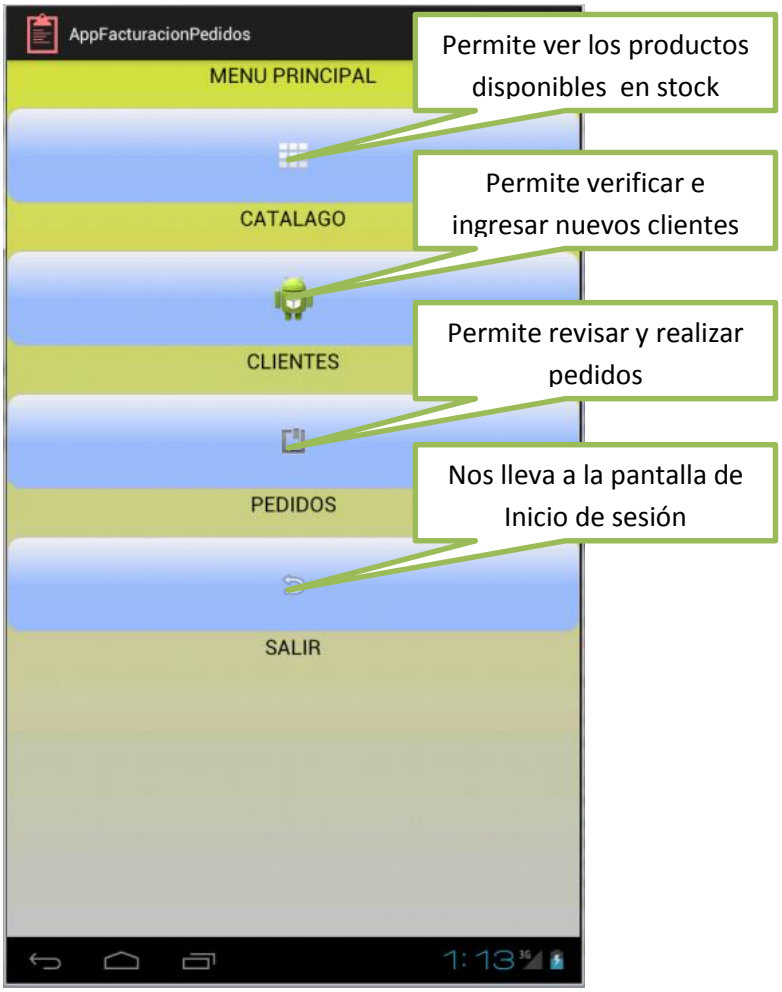


Figura 30. Luego de validar el ingreso como usuario se muestra la siguiente interface donde se puede escoger 4 diferentes opciones.

1.04. Catálogo

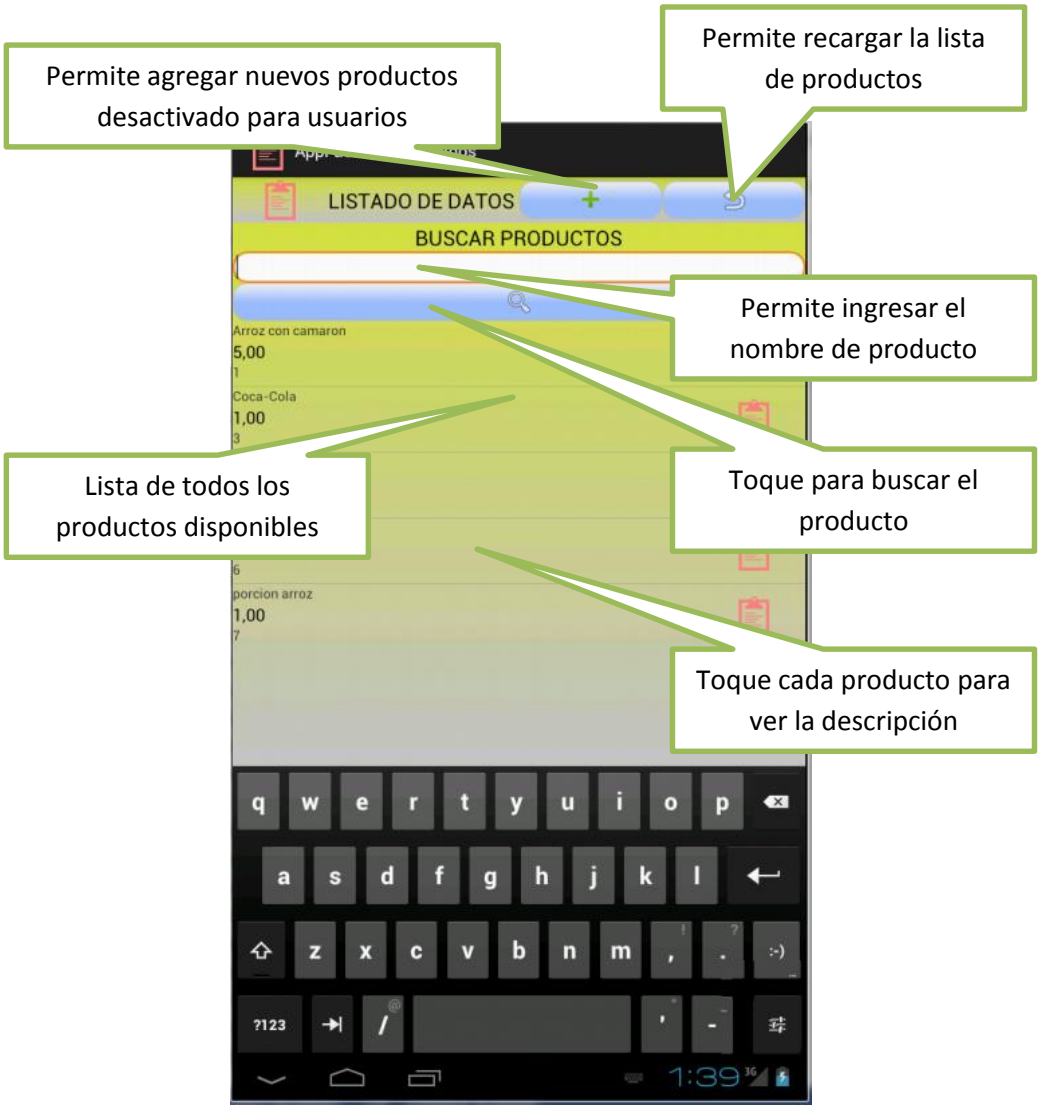


Figura 31. Permite revisar los productos que ofrece la empresa, realizar búsquedas por el nombre y ver la descripción completa de los productos.

1.05. Descripción de productos

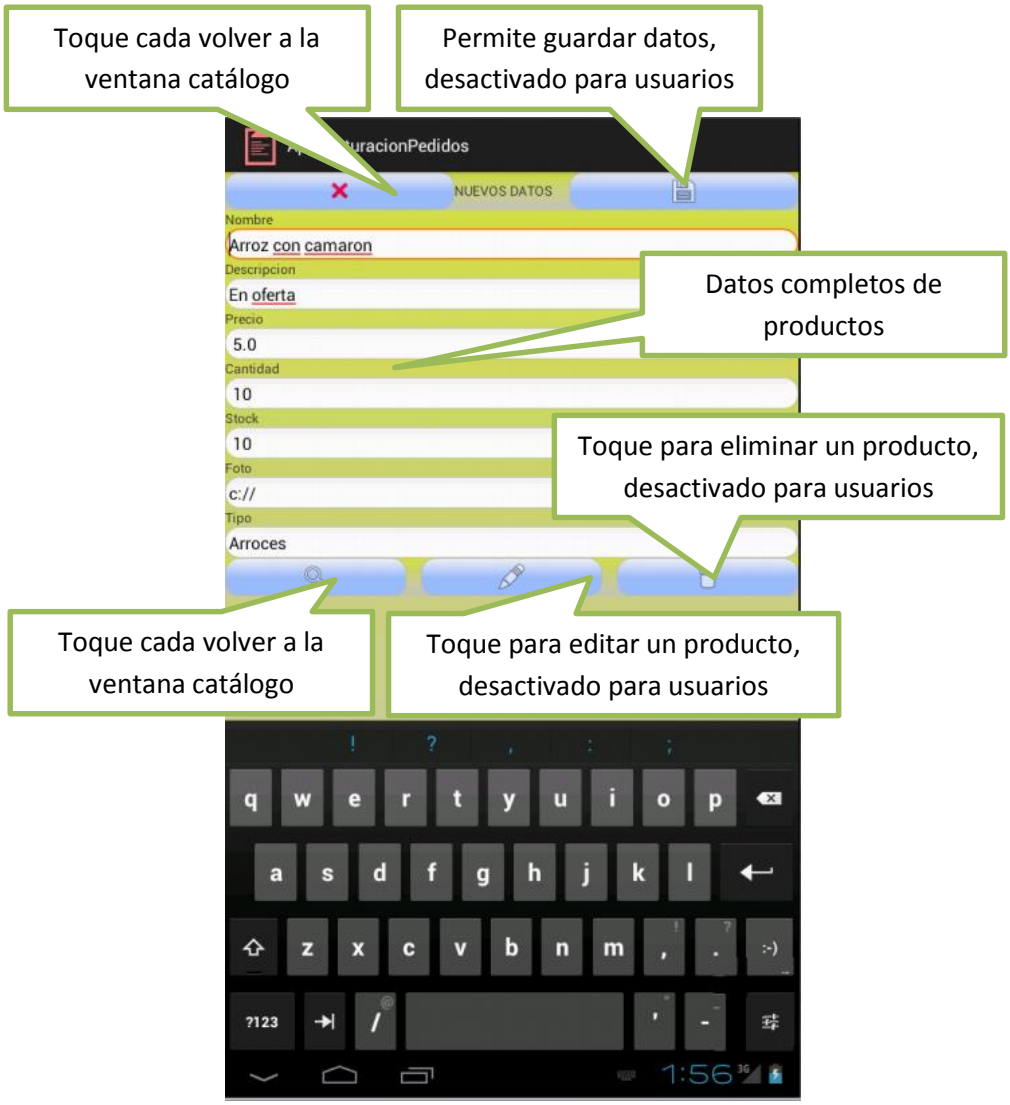


Figura 32. Esta ventana muestra una descripción completa de los productos las opciones de agregar, editar y eliminar está desactivadas para los usuarios y activas para los administradores.

1.06. Clientes

Figura 33. Muestra una lista completa de clientes registrados, además permite añadir, editar y recargar los datos

1.07. Descripción de clientes

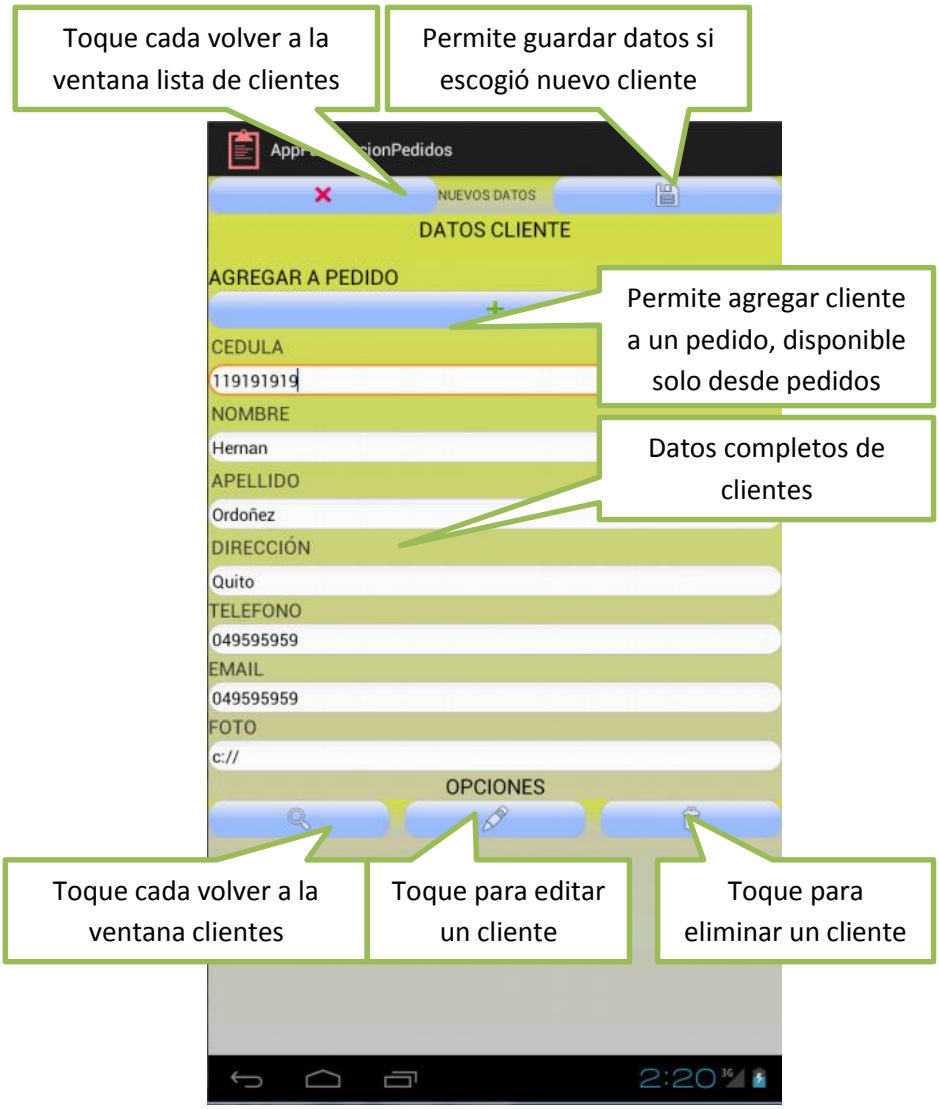


Figura 34. Esta ventana muestra una descripción completa del cliente seleccionado, existe opciones de agregar, editar y eliminar.

1.08. Pedido

Figura 35. Muestra una lista de todos los pedidos realizados, por el usuario que inicio sesión, permite buscar por número de mesa, agregar nuevo pedido y editar uno existente

1.09. Descripción de los pedidos

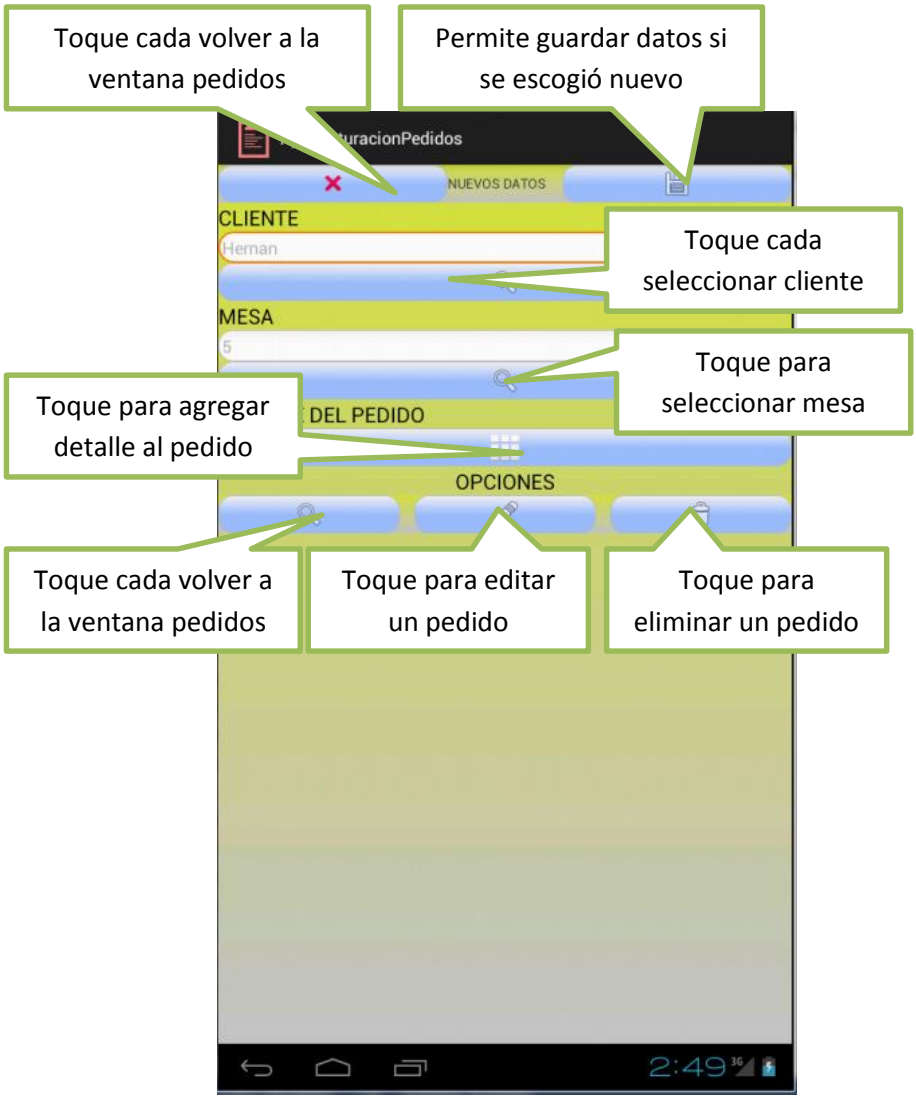


*Figura 36.*Aquí se muestra el cliente y la mesa del pedido, además permite crear, editar, eliminar y agregar el detalle de los pedidos.

1.10. Detalle Pedidos

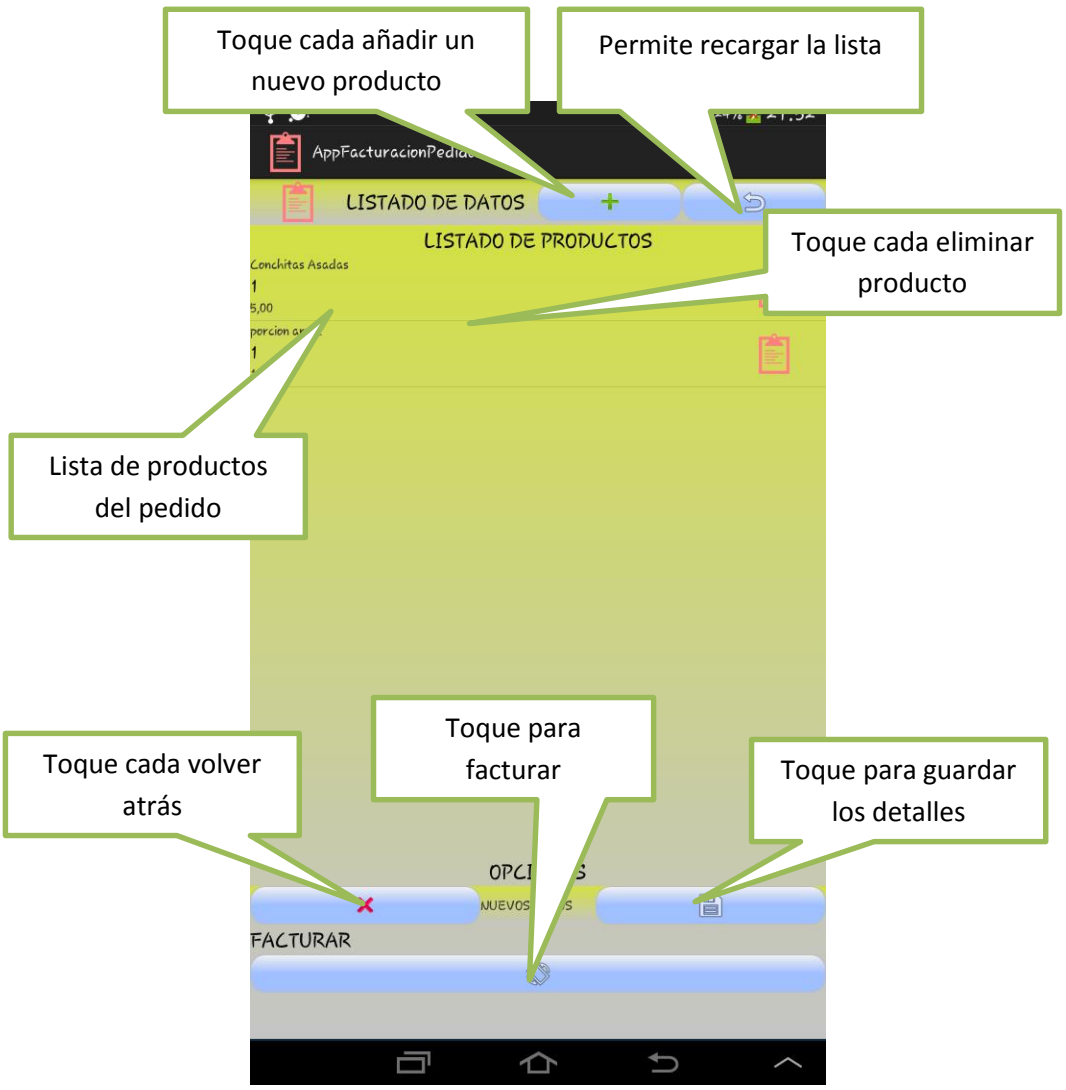


Figura 37. Menú de opciones que permiten realizar el detalle de los pedidos, y finalmente realizar la facturación.

1.11. Facturas.

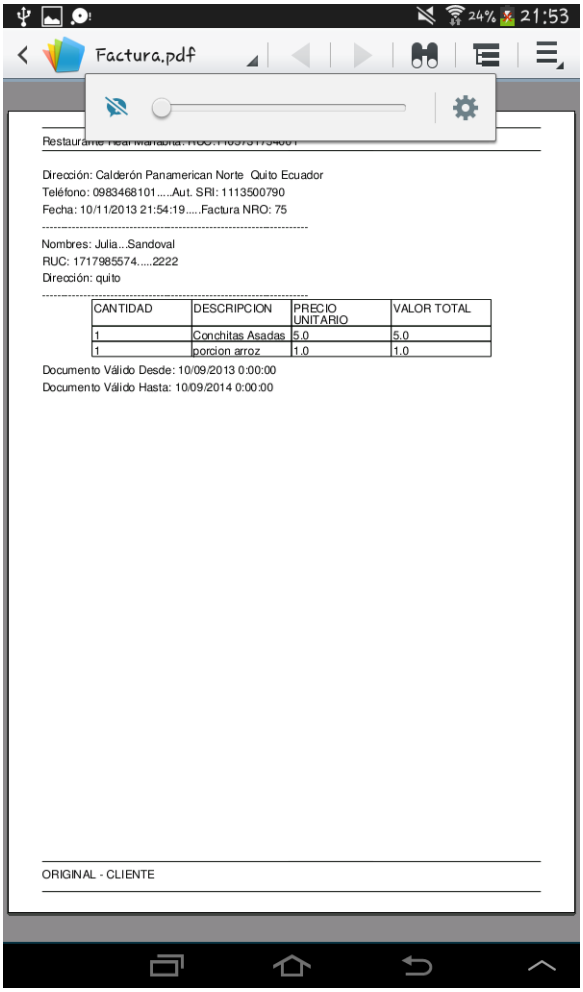


Figura 38. Las facturas generadas deberán abrirse con un lector de archivos PDF, para su correcta visualización.

1.12. Menú de administrador

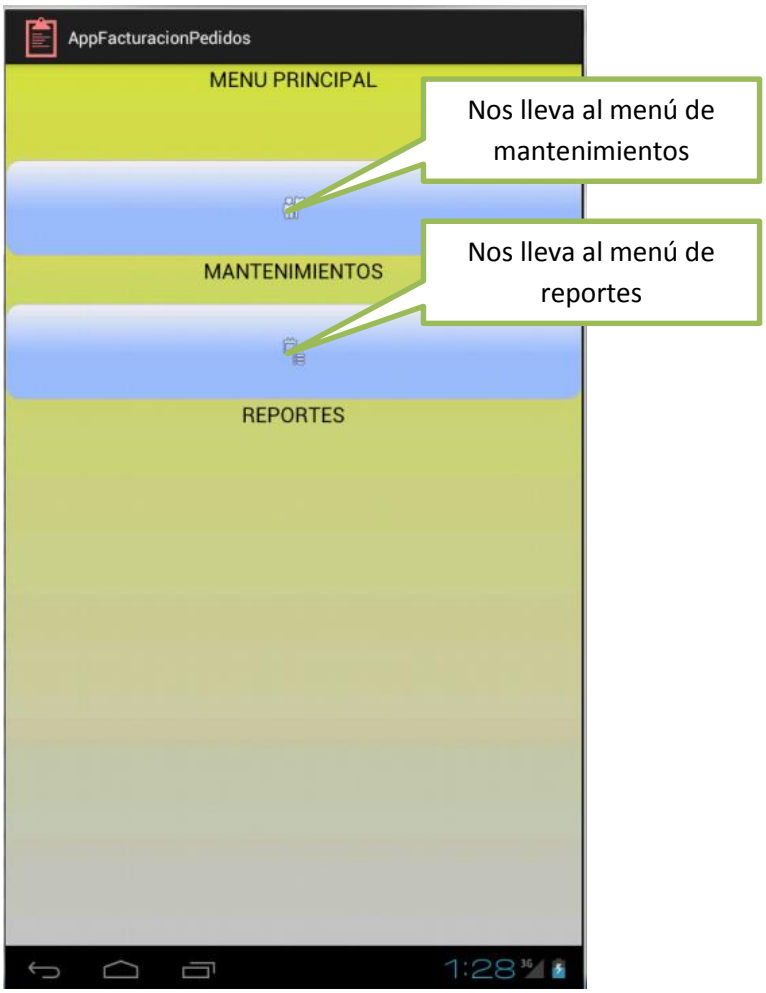


Figura 39. Validado correctamente el usuario como administrador, se muestra el menú para revisar reportes y realizar mantenimientos.

1.13. Menú de mantenimientos

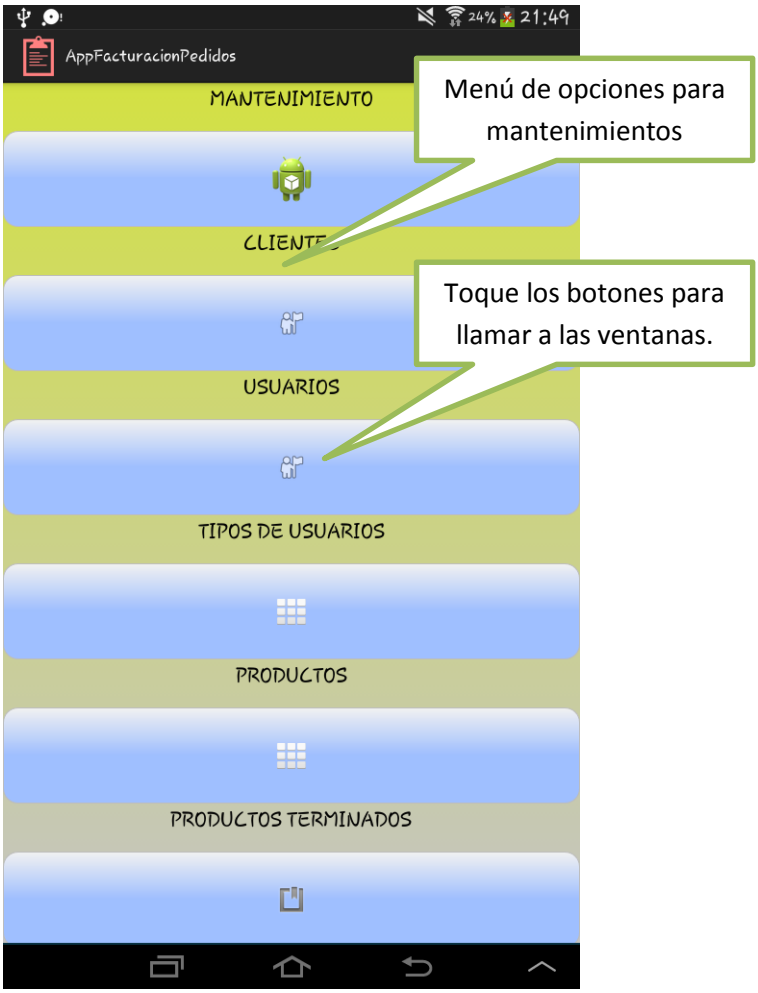


Figura 40. El menú de mantenimientos permite al administrador, realizar cambios en todas las tablas de la base de datos, y además verificar todos los procesos que se están llevando a cabo.

A.011. Manual de instalación.

El siguiente manual, tiene como objetivo mostrar al usuario como realizar la instalación de la aplicación, para su correcto funcionamiento.

1.01. Descarga de la aplicación

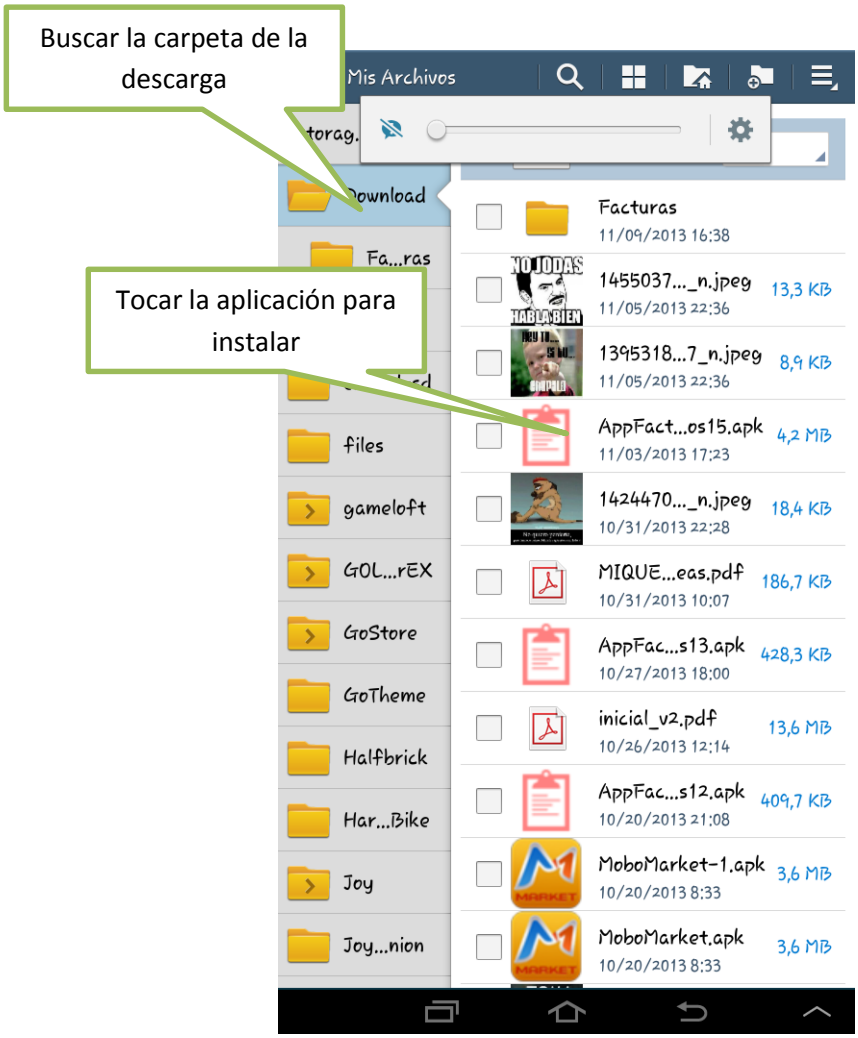


Figura 41. Como primer paso nos descargamos la aplicación, luego la ubicamos donde se descargó, en este caso en descargas, y hacemos clic sobre ella para instalar.

1.02. Instalar la aplicación.

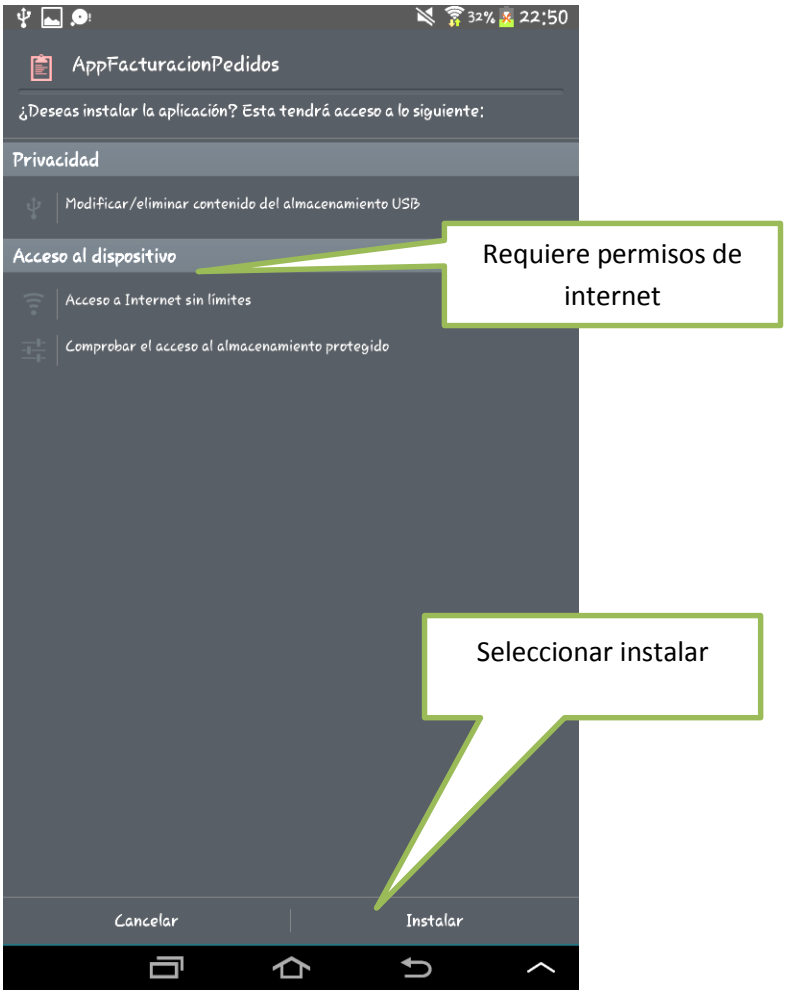


Figura 42. Luego de seleccionar la aplicación para instalar, nos mostrará un mensaje de confirmación de permisos, para la instalación, presionamos permitir y finalmente instalar, con lo que tendremos la aplicación en nuestro menú de la Tablet listo para su utilización.

A.012. Bibliografía.

Faustino Alarcón Valero, Ángel Ortiz Bas, María del Mar Eva Alemany Díaz, F. C. L. E. (2005). "Order promising" y Gestión de Pedidos: una visión de procesos. Retrieved December 10, 2012, from http://www.adingor.es/Documentacion/CIO/cio2005/items/ponencias/43.pdf

José Mª Vallsmadella. (2010). Gestion restaurantes. - El módelo básico de toma de decisiones en la gestión de restaurantes. Artículos e informes. Artículos de gestión. *08-01-2010*. Retrieved January 23, 2013, from http://www.gestionrestaurantes.com/llegir_article.php?article=448

Empresa ARAGON. (n.d.). Manual de Consulta Gestión de Pedidos y Distribución. Retrieved December 10, 2012, from http://www.programaempresa.com/empresa/empresa.nsf/0/e88d210e51f9371ac125705b002c66c9/$FILE/pedidos1y2.pdf

Crece Negocios. (2011). La toma de decisiones. Retrieved December 10, 2012, from http://www.crecenegocios.com/la-toma-de-decisiones/

Turismo, F. D. E., & Gastronomía, H. y. (2007). carrera de gastronomía tema :hatillo, e. l., miranda, e., peralta, s., & Alejandro, e. (2009). Para la creación de un restaurante de comida sushi-thai con ambiente "Premium" en el municipio trabajo especial de grado especialista en gerencia de proyectos asesor.

AUSIN. Edificios Inteligentes; Sistema de Administración de Energía SAE. Referencia Técnica. Catalogo comercial.

CozProf. Gestión de Restaurantes. [En línea] http://gestaoderestaurantes.wordpress.com/2008/05/04/pesquisa-sobre-automao-em-restaurantes/.

Editores 2009, *Automatización en restaurantes* Cocina Professional, págs. 14,15,16.

Electricidad Paredes Catálogo Comercial bajado de Internet en el Website http://www.dispal.com/El_Faro/Parades/simon2.htm

Garcia M., E (2001). Automatización de procesos industriales. México D. F: Alfaomega. Universidad Politécnica de Valencia.

Garcia Zubia, Javier y Martinez Angulo, Ignacio (2007) *temas digitales y cronología de computadoras.* Madrid: Tomson Editor es Spain

García, Emilio. *Automatización de Procesos Industriales s* Mexico DF. : Alfaomega, 2001.

Gonzales G., I (2007) Técnicas y procesos en las instalaciones singulares en los edificios. Madrid: 2da.Ed.