UNIVERSIDAD POLITÉCNICA SALESIANA

SEDE CUENCA

CARRERA DE INGENIERÍA DE SISTEMAS

Tesis previa a la obtención del Título de Ingeniero de Sistemas

TÍTULO:

"DISEÑO E IMPELMENTACIÓN DE UNA APLICACIÓN MÓVIL PARA TRABAJO OPERATIVO DE LOS VENDEDORES DE LA EMPRESA AGROTA CIA. LTDA."

AUTORES:

Johanna Nataly Picón Cajamarca

Sebastián Zhinin Zhinin

DIRECTOR:

Ing. Cristian Timbi

Cuenca, Diciembre de 2014

ECUADOR

Breve reseña del autor e información de contacto

Sebastián Zhinin Zhinin
Estudiante de la Carrera de Ingeniería de Sistemas
Universidad Politécnica Salesiana
szhinin89@gmail.com

Johanna Nataly Picón Cajamarca
Estudiante de la Carrera de Ingeniería de Sistemas
Universidad Politécnica Salesiana
jhoapicon17@gmail.com

CERTIFICA

Haber dirigido y revisado prolijamente cada uno de los capítulos del informe de tesis realizado por el Sr. Sebastián Zhinin Zhinin y la Srta. Johanna Nataly Picón Cajamarca, y por cumplir los requisitos autorizo su presentación.

Cuenca, Diciembre del 214

Ing. Cristian Fernando Timbi Sisalima

Director de Tesis

DEDICATORIA DE RESPONSABILIDAD

Nosotros, Sebastián Zhinin Zhinin portador de la cédula de ciudadanía 0302126842 y Johanna Nataly Picón Cajamarca portadora de la cédula de ciudadanía 0105836183, estudiantes de la Carrera de Ingenierías de Sistemas, certificamos bajo juramento que los conceptos, contenido y nociones desarrollados son de exclusiva responsabilidad de los autores, y que hemos consultado la referencias bibliográficas que se incluyen en el presente documento.

A través de la presente declaración cedemos nuestros derechos de propiedad intelectual correspondientes a este trabajo, a la Universidad Politécnica Salesiana, según lo establecido por la Ley de Propiedad Intelectual, por su reglamento y por la normativa institucional vigente.

Cuenca, Diciembre del 2014

Sebastián Zhinin Zhinin

Johana Nataly Picón Cajamarca

DEDICATORIA

A mis padres

Fausto Picón y Cecilia Cajamarca por ser el pilar fundamental de mi vida, por todo el cariño y comprensión que he recibido de ustedes, por todos los esfuerzos que hicieron para que lograra culminar mi meta y sobre todo gracias el amor incondicional que solo unos padres lo pueden dar.

A mi querida tía Sonia Beatriz Roldán Cajamarca

Por ser como mí segunda madre, porque a pesar que estas lejos siempre con tus palabras de ánimo me enseñaste a ser perseverante y no dejarme derrotar por las circunstancias difíciles, por ser mi ejemplo a seguir esta tesis va dedicada para ti mi querida tía en agradecimiento a todo el cariño que he recibido de tu parte.

Al amor de mi vida

Por ser mí apoyo incondicional, por haber compartido tantos buenos y malos momentos, por motivarme cada día a ser mejor persona y sobre todo por haber creído en mi porque a pesar de que los ánimos se me iban siempre encontrabas las palabras correctas para sacarme una sonrisa, por todo lo dicho y mucho más esta tesis es para ti.

Johanna Picón

DEDICATORIA

La tesis le dedico a Dios que ha estado presente en todo momento en cada paso que doy, a mi familia especialmente a mis padres María Zhinin Guamán y Bernardo Zhinin Álvarez, quienes me motivaron seguir adelante para llegar a la meta, y seguir adelante con nuevos objetivos.

A mi hermano José Antonio Zhinin junto con su esposa Zoila Chimbo que siempre han estado pendiente de mis estudios, y aportando moralmente y económicamente durante toda mi carrera universitaria, brindándome sus consejos en los momentos que lo he necesitado y a mis sobrinos José, Sara y Antonio.

Sebastián Zhinin

AGRADECIMIENTO

Agradezco a Dios por la vida, la salud y por haberme dado la capacidad suficiente para poder alcanzar esta meta, porque siempre me condujo por el bueno camino y no me desamparo en ningún momento.

A mis queridos tíos Iván Picón, Geovanny Picón, Aida Picón por siempre apoyarme con sus consejos, por compartir conmigo sus experiencias y permitir que yo aprenda de ellas. También deseo extender mis agradecimientos a primas Marisol Zumba y Paola Picón por enseñarme apreciar la vida, pequeños detalles que a veces no vemos cuando tenemos cerca a la familia y los aprendemos a valorar cuando se encuentran lejos. Gracias a toda mi familia, hermanos y primos por incentivarme a salir adelante a lo largo de toda mi vida.

Mi más sincero agradecimiento a mi compañero de tesis por todo el apoyo brindado durante la realización de este trabajo, por compartir sus conocimientos, por ayudarme siempre a salir adelante y enseñarme a confiar más en mí. Asimismo quiero agradecer la amistad de cada uno de los amigos que encontré durante mi vida universitaria y que hoy me permito compartir con todos ustedes esta gran alegría

También agradezco a nuestro tutor de tesis el Ing. Cristian Timbi que con su experiencia y generosidad nos ha brindado la confianza para culminar con éxito el presente trabajo. A su vez quiero agradecer a los distinguidos docentes de la Universidad Politécnica Salesiana al Ing. Byron Carrión, Ing. Pablo Gallegos, Ing. Rodolfo Bojorque y al Ing. Jairo Sacoto por su vocación, confianza y amistad.

Johanna Picón

AGRADECIMIENTO

Primer lugar a Dios por ser mi guía incondicional en todo momento de mi vida, por enseñar el camino y las decisiones correctas y vivir este momento que ha sido uno de las más grandes metas, a mi madre María Zhinin Guamán gracias por el apoyo que me ha dado, a mi padre Bernardo Zhinin Álvarez por la fuerza y consejos que me ha brindado para seguir adelante. A mi hermano José Antonio Zhinin junto con su esposa Zoila Chimbo que es uno de los principales apoyo que he tenido en todo el camino de vida estudiantil para poder cumplir mi meta y ser mejor persona. Gracias a todo ellos por creer en mí.

Agradezco a mis hermanos Bernardo y Toribia que a pesar de distancia me apoyado, para seguir mejorando ya como persona y como estudiante, gracias por todo.

A mi compañera de tesis gracias por ayudar a tomar decisiones correctas y brindar el apoyo, que ha sido fundamental para que los proyectos y los trabajos salgan bien en los últimos años de estudiante.

A nuestro tutor de tesis el Ing. Cristian Timbi que con su experiencia y conocimiento nos ha brindado la confianza para culminar con éxito el presente trabajo. A los docentes de la Universidad Politécnica Salesiana al Ing. Byron Carrión, Ing. Pablo Gallegos y Ing. Rodolfo Bojorque.

Agradezco a la empresa Agrota Cía. Ltda. Por habernos permitido realizar la tesis para culminar nuestra etapa universitaria. En particular al Ing. Francisco Maldonado quien gracias su apoyo se pudo concluir el presente trabajo de tesis.

Sebastián Zhinin

ÍNDICE GENERAL

Capítulo 1

Capítulo 3

Capítulo 4

ÍNDICE DE ILUSTRACIONES

ÍNDICE DE TABLAS

CAPÍTULO 1

1.NUEVAS TECNOLOGÍAS APLICADAS AL ÁREA DE VENTAS

1.1. Estudio de la Tecnología Android para dispositivos móviles

Actualmente existe una gran variedad de plataformas para dispositivos móviles (Iphone, Symbian, Windows Iphone, BlackBerry, Java Mobile, etc.), sin embargo Android se ha convertido en una herramienta poderosa para el desarrollo. Su gran aceptación se debe a la gran cantidad de aplicaciones gratuitas, flexibilidad, código abierto, entorno amigable. Android es el primero en consolidar en una misma solución las siguientes características:

- **Plataforma realmente abierta:** Es una plataforma de desarrollo libre basada en Linux y de código abierto.

- **Portabilidad asegurada:** Las aplicaciones desarrolladas podrán ser ejecutadas en diferentes dispositivos ya que son creadas bajo el concepto de Java.

- **Adaptable a cualquier tipo de hardware:** Android no está restringido solo al uso en teléfonos y tablet, la tendencia actual es la creación de accesorios tecnológicos tales como: gafas, cámaras, pulseras inteligentes, perdiéndonos ir más allá de lo que el lenguaje nos brinda.

- **Arquitectura basada en componentes inspirados en Internet:** Promueve la estandarización mediante XML, permitiendo que cualquier aplicación Android puede ejecutarse en un móvil de pantalla reducida [1].

- **Variedad de servicios incorporados:** Compuesto por una poderosa base de datos SQL, permite incorporar navegadores, localización mediante GPS, multimedia, síntesis de voz directamente a la aplicación [1,2].

- **Seguridad:** Al emplear el concepto de Linux todos los programas se encuentran aislados como si estuvieran dentro de una caja, sin embargo toda aplicación creada debe manejar un modelo de permisos para garantizar la integridad de los datos [2].

- **Alta calidad de gráficos y sonidos:** Gráficos vectoriales y 3D, animaciones en flash, soporte para archivos MPG4, H.264, MP3, AACC, JPG, PNG, GIF, etc. [1].

1.1.1 Comparativa frente a otras plataformas

El Kernel es el corazón de un sistema, es un software que constituye la parte más importante del sistema operativo. *"Es el principal responsable de facilitar a los distintos programas acceso seguro al hardware de la computadora o en forma básica, es el encargado de gestionar recursos, a través de servicios de llamada al sistema"* [3]. En la Ilustración 1 expondremos los aspectos más destacados al comparar las diferentes plataformas para móviles:

- ✓ **Licenciamiento:** El software propietario representa una gran limitante para la creación de nuevas aplicaciones mientras que el software libre ofrece libertad para crear, modificar o desarrollar nuevas funcionalidades.

- ✓ **Costo:** Si bien un código abierto representa costos iniciales altos, el desarrollo resulta ser más fácil. En una opción cerrada el alto costo estará presente a lo largo de todo el proyecto y el desarrollo estará limitado a quien creo el código.

- ✓ **Número de aplicaciones:** Android contiene alrededor de 850.000 aplicaciones la mayoría gratuitas siendo una opción atractiva tanto para usuarios como para desarrolladores.

- ✓ **Variedad de dispositivos:** La segmentación de Android ha permitido la creación de diferentes versiones tanto para tablet como teléfonos. En IOS las versiones están restringidas para un dispositivo móvil único.

	Apple iOS 7	Android 4.3	Windows Phone 8	BlackBerry OS 7	Symbian 9.5
Compañía	Apple	Open Handset Alliance	Microsoft	RIM	Symbian Foundation
Núcleo del SO	Mac OS X	Linux	Windows NT	Mobile OS	Mobile OS
Licencia de *software*	Propietaria	Software libre y abierto	Propietaria	Propietaria	Software libre
Año de lanzamiento	2007	2008	2010	2003	1997
Fabricante único	Sí	No	No	Sí	No
Variedad de dispositivos	modelo único	muy alta	media	baja	muy alta
Soporte memoria externa	No	Sí	Sí	Sí	Sí
Motor del navegador web	WebKit	WebKit	Pocket Internet Explorer	WebKit	WebKit
Soporte Flash	No	Sí	No	Si	Sí
HTML5	Sí	Sí	Sí	Sí	No
Tienda de aplicaciones	App Store	Google Play	Windows Marketplace	BlackBerry App World	Ovi Store
Número de aplicaciones	825.000	850.000	160.000	100.000	70.000
Coste publicar	$99 / año	$25 una vez	$99 / año	sin coste	$1 una vez
Actualizaciones automáticas del S.O.	Sí	depende del fabricante	depende del fabricante	Sí	Sí
Familia CPU soportada	ARM	ARM, MIPS, Power, x86	ARM	ARM	ARM
Máquina virtual	No	Dalvik	.net	Java	No
Aplicaciones nativas	Siempre	Sí	Sí	No	Siempre
Lenguaje de programación	Objective-C, C++	Java, C++	C#, muchos	Java	C++
Plataforma de desarrollo	Mac	Windows, Mac, Linux	Windows	Windows, Mac	Windows, Mac, Linux

Ilustración 1. Comparativa de las principales plataformas móviles [4].

Analizando la Ilustración 1 observamos que IOS posee un código muy estable con cada nueva versión ofrece mejoras significativas con un excelente acabado tanto estético como funcional. Por su parte Windows Phone y BlackBerry brindan una interfaz amigable e intuitiva, sin embargo su tienda de aplicaciones cuenta con valores muy bajos, además que la mayoría no se pueden probar sin antes comprarlas. En cuanto a Symbian se considera una plataforma obsoleta que tiene poco que aportar y está desapareciendo del mercado. Android por su parte ofrece sencillez a la hora de programar, publicación instantánea, multitarea, personalización, alta gama de dispositivos, bajo costo, etc. Estas características lo han convertido en un sistema operativo multifunción y completamente escalable.

Desde otro punto de vista analizando la evolución del mercado de los sistemas operativos para móviles según el número de terminales vendidos. Podemos destacar el importante descenso de ventas de la plataforma Symbian de Nokia, el declive continuo de BlackBerry, el descenso de Apple en el mercado. Finalmente destacamos el espectacular ascenso de la plataforma Android, que le ha permitido alcanzar en dos años una cuota de mercado superior al 75% [4]. Ilustración 2.

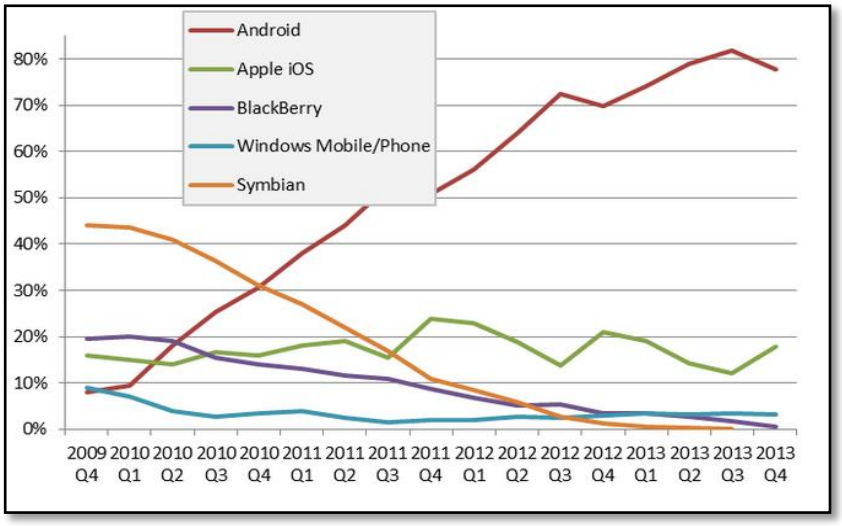


Ilustración 2.Porcentaje de teléfonos inteligentes vendidos según su sistema operativo hasta el último cuarto del 2013 en el mundo [5].

Dada la situación del mercado mundial Android es la plataforma dominante y la preferida por programadores ya que no solo nos permite modificar el código fuente sino también mejorarlo de acuerdo a la necesidad.

Si hablamos de IOS como un gran perdedor lo tratamos en el contexto de su bajo porcentaje de ventas y descarga de apps. Muchas aplicaciones IOS nunca han sido descargadas y nadie ha descubierto su funcionalidad, son aplicaciones que han pasado desapercibidas en el catálogo.

Un ejemplo son los 1.899 programas linterna de IOS, la mayoría hacen lo mismo excepto algunos que destacan una ventaja clave. Según el estudio *Adeven*, Apple cuenta con 400.000 aplicaciones fantasma las cuales nunca fueron descargadas. Definitivamente la compañía debería considerar una nueva forma de destacar sus aplicaciones [32].Ilustración 3.

Ilustración 3. Aplicaciones linterna de un Iphone [32].

Hoy por hoy existen una gran variedad dispositivos móviles suficientemente pequeños, resistentes y portables. Los dispositivos Android han demostrado ser los líderes en el mercado ya que sus diseños van más allá de un hardware potente incorporando nuevas funcionalidades como ahorro de batería extremo, sensores dactilares, sensores de profundidad, etc. [6][7].

Ilustración 4. Los mejores Smartphone del Mobile Word Congress 2014 [8].

Marcas líderes en el mercado como Nokia, Samsung, Sony han apostado por Android viéndolo como una fuerza imparable que vende. La elección del sistema operativo por parte de un fabricante, representan una serie de estrategias para ampliar el mercado, atraer clientes, crear nuevas tendencias, etc.

1.1.2 Arquitectura de Android

Otro aspecto fundamental a considerar es la arquitectura modular de Android la cual es realmente escalable, robusta y sobre todo portable permitiéndonos asegurar que las aplicaciones desarrolladas podrán ser ejecutadas en una variedad de dispositivos tales como tablet, notebook, PDAs, Smartphone, etc. Un dispositivo Android está formado por la siguiente estructura. Ilustración 5.

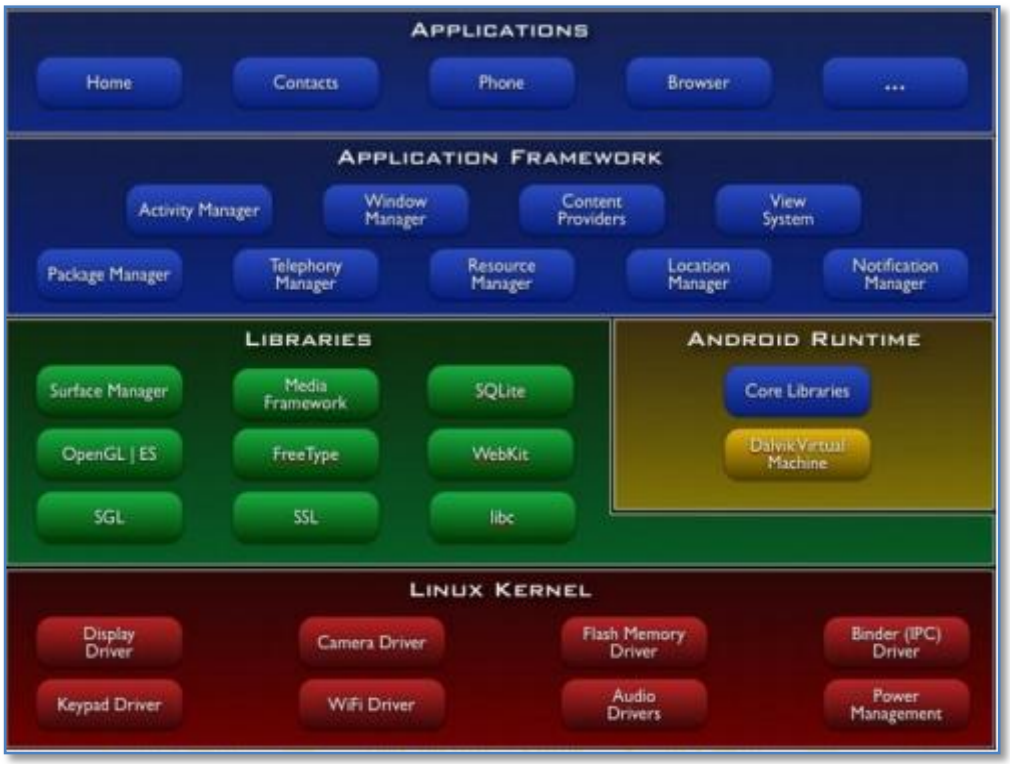


Ilustración 5.Arquitectura de Android [10].

1.1.2.1 El núcleo de Linux

Está basado en software libre por lo tanto los servicios base del sistema como: la seguridad, la gestión de memoria, la gestión de proceso, pila de red y el modelo de controladores son gestionadas por las librerías del nucleó de Linux. ‘‘*Funciona como capa de abstracción entre el hardware y el resto de la pila, por lo tanto es la única que es independiente del hardware*'' [1].

1.1.2.2 Runtime de Android

Basado en el concepto de máquina virtual Dalvik, incluye la ejecución de ficheros Dalvik Execute (.dex), una extensión que optimiza el almacenamiento y ejecución mapeable en memoria. También está compuesto por registros, cada aplicación corre en su propio proceso Linux con su propia instancia de la máquina virtual.

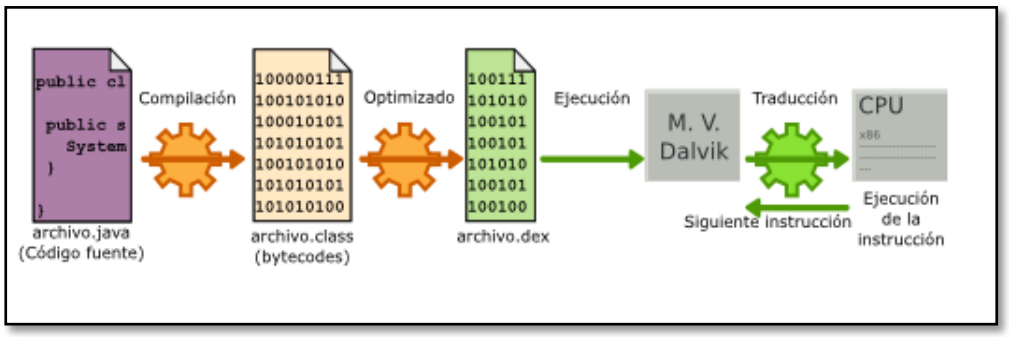


Ilustración 6. Engranes que mueven a Dalvik [28].

La Ilustración 6 explica el funcionamiento de Dalvik. Aquí las aplicaciones son creadas en Java, mediante un compilador transformamos el código Java a un código binario. La máquina virtual interpreta este bytecode generando un archivo .dex, el cual será interpretado por la máquina virtual Dalvik y traducido a código entendido por el CPU [28].

1.1.2.3 Librerías nativas

Incluye una gran variedad de bibliotecas en C/C++ usadas por ciertas partes del sistema, los desarrolladores pueden hacer uso de ellas de acuerdo a su necesidad.

- **Surface Manager:** Gestiona el acceso a la pantalla a partir de capas gráficas 2D y 3D.
- **Media Framework:** Reproducción de imágenes, audio y video en una gran variedad de formatos: MPEG4, H.264, MP3, AAC, AMR, JPG y PNG [1].
- **SQLite:** Es una pequeña base de datos relacional.
- **Open GL | ES:** Representa las librerías gráficas y por lo tanto sustentan la capacidad gráfica de Android

- **FreeType:** Permite trabajar de forma rápida y sencilla con vectores e imágenes.
- **WebKit:** Moderno navegador web utilizado en el navegador Android.
- **SGL:** Motor de gráficos 2D [1].
- **SSL:** Proporciona servicios de encriptación Secure Socket Layer [1].

1.1.2.4 Entorno de la aplicación

Incluye el coreLibraries de Android basado en el lenguaje Java, diseñado justamente para facilitar la reutilización de componentes y promover el desarrollo libre de nuevas aplicaciones. A nivel de framework de aplicación encontramos algunas de las librerías más importantes:

- **Activity Manager:** Administra y maneja el ciclo de vida de las aplicaciones.

- **TelephoneManager:** Incluye los APIs vinculados a la funcionalidad propia del teléfono (mensajes, llamadas).

- **Resource Manager:** Gestiona todos los elementos que forman parte de la aplicación y están fuera del código. Ejemplo: cadenas de texto traducidas a diferentes idiomas, imágenes, sonidos o layouts [1].

- **Views System**: Son los elementos que componen la interfaz visual para el usuario.

- **Content Providers:** Permite a una aplicación compartir sus datos con otras aplicaciones Android, gracias a este API la información de contacto, mensajes podrá ser accesible desde otras aplicaciones [1].

- **Notification Manager:** permite a las aplicaciones mostrar alertas personalizadas en la barra de estado.

1.1.2.5 Aplicaciones Android

Es la capa superior de la pila de software, incluye todas las aplicaciones del dispositivo tanto las que tienen interfaz de usuario, aplicaciones nativas, las que vienen por defecto o las instaladas por el usuario [2]. A continuación se presenta los componentes básicos de una aplicación:

- **Activity:** Un componente Activity representa el ciclo de vida de una aplicación y está asociado a una ventana o interfaz de usuario.

- **BroadcastIntent Receiver:** Son tareas que están constantemente ejecutándose esperando reaccionar ante un determinado evento (batería baja, mensajes).

- **Service:** Son componentes sin interfaz gráfica que no requieren la interacción con el usuario, se ejecutan en segundo plano cuando alguna acción esta activa. Ejemplo: notificaciones, actualización de datos, etc.

- **Content Provider:** Es un mecanismo estándar que permite recuperar, actualizar y compartir datos ya sea mediante archivos o a través de la base de datos SQLite [1,2].

Las aplicaciones móviles están dirigidas por eventos implicando que el desarrollador aprenda a controlar los diferentes estados que puede tomar una actividad.

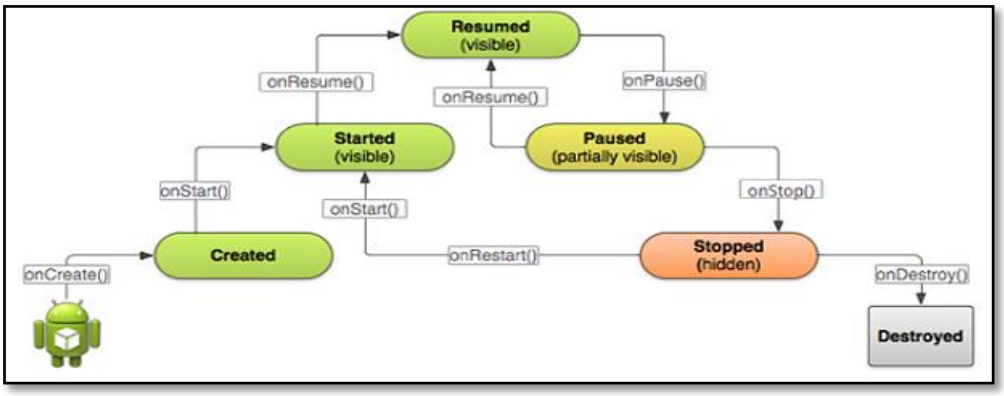


Ilustración 7. Ciclo de vida de un componente Activity [29]

En la lustración 7 analizaremos el ciclo de vida de una actividad, los estados se encuentran representados por óvalos mientras que los métodos por el prefijo "on". Para llegar a un estado debemos pasar por una serie de métodos. Ejemplo: Para llegar al estado ***Created*** de una actividad, pasamos por el método ***onCreate().*** Para pasar al estado ***Started*** desde el estado ***Stopped*** pasamos por los métodos ***onRestart ()*** y ***onStart ()*** [24]. En el siguiente cuadro describiremos los diferentes métodos que puede tener una actividad.

Estado	Descripción	Siguiente
onCreate()	Es el encardo de inicializar la aplicación y se ejecuta una sola vez.	onStart()
onRestart()	Se ejecuta después que la aplicación se detuvo, justo cuando la actividad se está volviendo a mostrar al usuario.	onStart()
onStart()	Se ejecuta antes que la aplicación sea visible para el usuario. Si la aplicación es un proceso en background el siguiente método es onStop(), si la aplicación se ejecuta en foreground el siguiente método es onResume().	onResume() onStop()
onResume()	Se usa cuando la aplicación está ejecutando e interactuando con el usuario.	onPause()
onPause()	Se ejecuta cuando la aplicación está a punto de continuar con una actividad anterior. Seguida por onResumen() si la actividad es visible para el usuario o onStop si no es invisible.	onResume() onStop()
onStop()	Se ejecuta cuando la actividad es invisible para el usuario. Él siguiente método será onRestart() si la actividad vuelve a interactuar con el usuario o onDestroy() si la actividad fue destruida.	onRestart() onDestroy()
onDestroy()	Es la última llamada antes de destruir la Activity, puede ocurrir porque la actividad está acabando o porque el sistema destruye la instancia para ganar espacio.	FIN

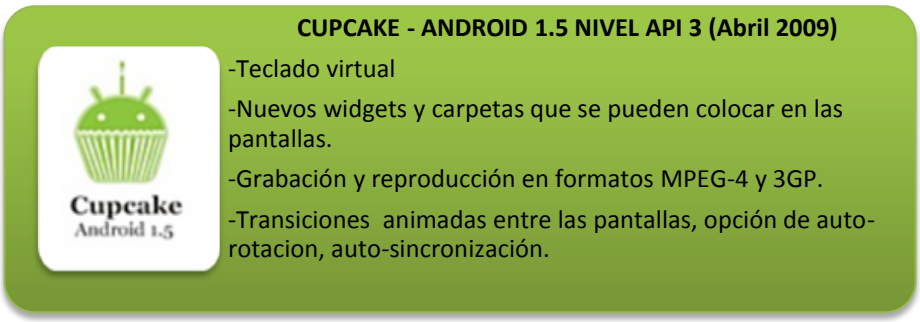
Tabla 1. Descripción de los métodos de un Activity [30].

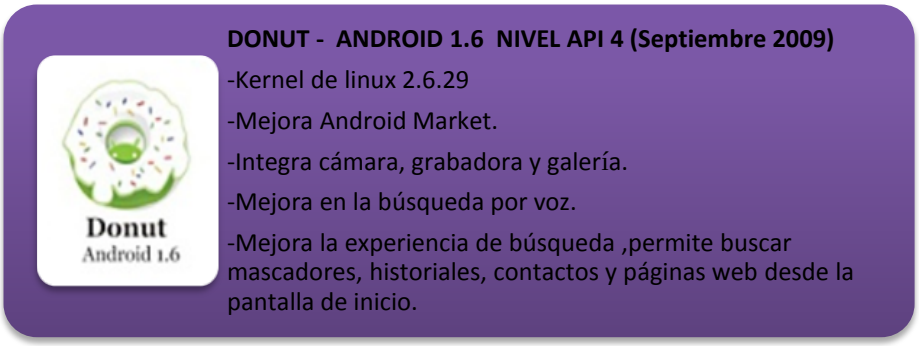
1.1.3 Versiones Android y Niveles API

Antes de comenzar a desarrollar un proyecto en Android es importante analizar qué versión vamos a utilizar ya que existen clases, métodos y características especiales que están disponibles a partir de una versión determinada. En la siguiente ilustración expondremos las plataformas lanzadas hasta la fecha con una breve descripción de las novedades incorporadas.

- **Android 1.0 Nivel de API 1 (septiembre 2008):** Fue la primera versión de Android y la que marco un nuevo ambiente de telefonía móvil basado en software libre. Entre las novedades más destacadas tenemos:
  - Incorporación de un mercado para compra y descarga de aplicaciones bajo el nombre de Android Market.
  - Soporte para cámara
  - Acceso a servidores de correo electrónico: POP3, IMAP4 y SMTP.
  - Mensajería instantánea.
- **Android 1.1 Nivel de API 2 (Febrero 2009):** No existen grandes mejoras a la versión anterior por lo que apenas existen usuarios con esta versión.

- **Android 1.5 Nivel de API 3 (Abril 2009):** Esta versión se basa en el núcleo de Linux 2.6.27 de aquí en adelante todas las versiones de Android están basadas en el mismo núcleo.





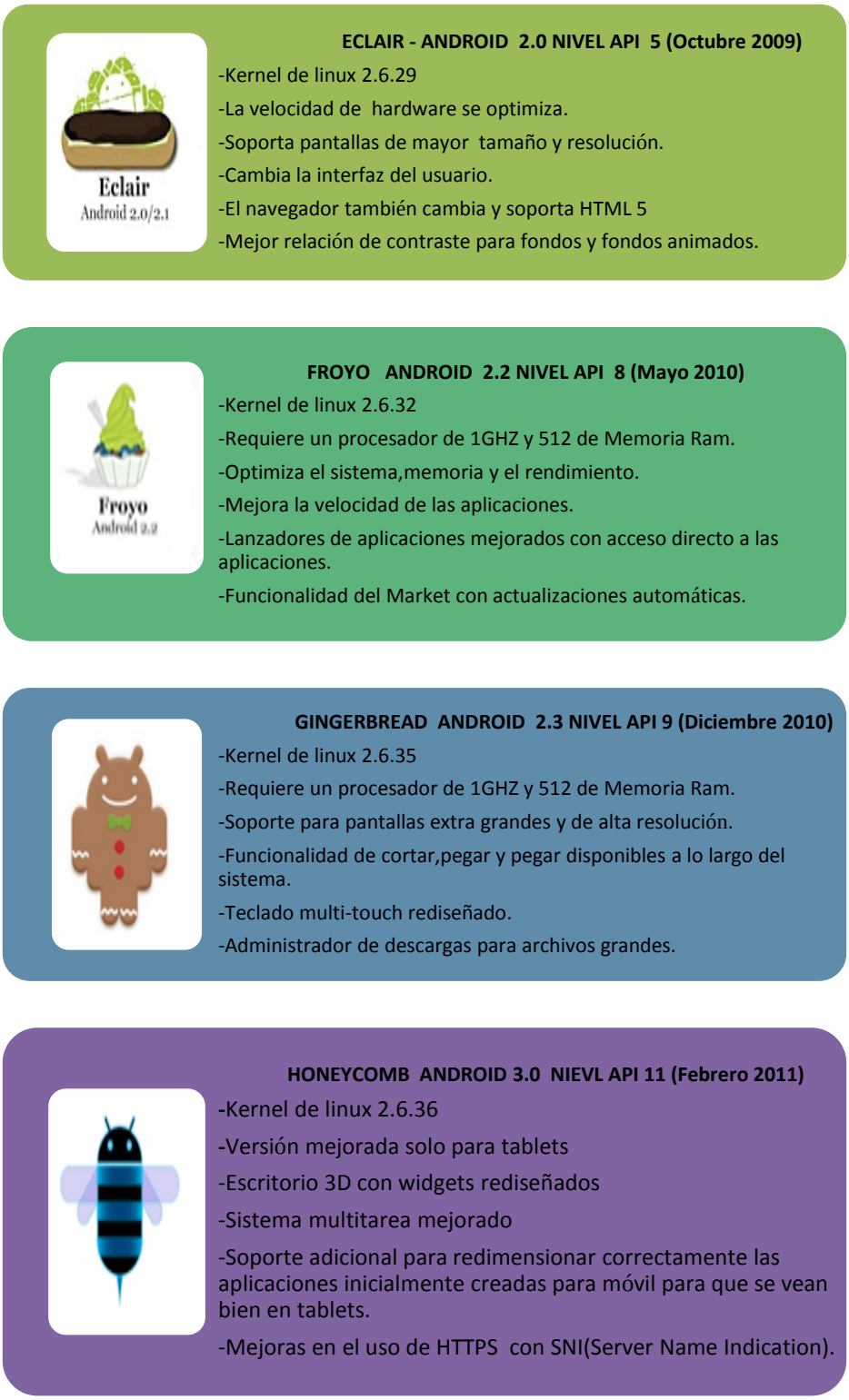


Ilustración 8. Versiones Android y Niveles API [1, 2,9].

Android 3.1 Nivel de API 12 (mayo 2011): Permite manejar dispositivos conectados por USB (tanto host como dispositivo).

Android 3.2 Nivel de API 13 (julio 2011): Optimizaciones para distintos tipos de tablet, zoom compatible para aplicaciones de tamaño fijo y sincronización multimedia

Ilustración 9. Versiones Android y Niveles API [1, 2, 9]

1.2 Estudio del entorno de Android en el área de ventas

1.2.1 Interacción con el vendedor

Actualmente se ha reestructurado la forma en que nos desenvolvemos en el ámbito laboral, ahora las tareas diarias pueden complementarse fuera de la oficina mediante el uso de dispositivos móviles.

Los vendedores pueden aprovechar la potencia y eficiencia de las aplicaciones Android para simplificar su trabajo operativo, organizar las tareas del negocio, evitar el deterioro y pérdida de documentos.

Entre las aplicaciones Android gratuitas más destacadas tenemos:

- **Agenda Calendar:** Facilita la creación de eventos de una forma rápida y sencilla.
- **Evernote:** Es una herramienta que sirve para guardar notas, escribir artículos, guardar documentos, dictado de voz, etc.
- **Stocks:** Permite gestionar las finanzas y cotizaciones con Android [33].
- **Documents To Go:** Herramienta que nos permite visualizar documentos en diferentes tipos de formato ya sea Word, Excel, Power Point, etc.
- **Google Voice:** Permite vincular un número de teléfono y llamar a distintos lugares o dispositivos diferentes [33].
- **Expense Manager:** Gestiona los gastos e ingresos de la empresa, además de bridar la posibilidad de programar pagos a efectuar [33].

No cabe duda que los procesos front end como la interfaz de usuario puede ser uno de los factores decisivos para el éxito o al fracaso de una aplicación. Por lo que la aplicación móvil representa para el vendedor una herramienta de apoyo para el registro de pedidos, consulta de catálogos, consulta de combos promocionales, consulta de clientes, etc.

1.2.2 Android y el acceso a la Base de Datos

El modelo de negocio de Agrota se ve influenciado por la tecnología de forma directa o indirecta. Cabe recalcar que esta influencia afecta desde los procesos más básicos inherentes en la lógica de negocio, tales como control de usuarios, acceso a la BD, servicios web, comunicación, sincronización, etc.

1.2.2.1 Control de acceso

Si bien la creación de multiusuarios se convirtió en un gran problema para Android con la versión Jelly Bean 4.3 la cual usaremos, se ha superado el inconveniente permitiéndonos crear perfiles restringidos y usarlos para varios fines. Podemos implementar los siguientes perfiles:

- **Quiosco:** permite configurar la tablet para realizar demostraciones de aplicaciones y funciones seleccionadas a los clientes [26].

- **Venta al público:** permite que los clientes puedan explorar las funciones de la tablets, pero sin que puedan explorar o utilizar otros servicios [26].

- **Punto de venta:** permite que los empleados solo puedan usar las aplicaciones seleccionadas de ventas y de registro [26].

Dependiendo la prioridad del usuario podemos crear los privilegios para ese perfil ya sea creando un usuario normal el cual tendrá acceso al contenido y aplicaciones propias o creando un perfil limitado restringido por el administrador.

1.2.2.2 Manejo de usuarios

Debemos asignar un rol de trabajo a cada usuario de esta forma podremos establecer los permisos necesarios para acceder a la aplicación. En nuestro caso los involucrados son: administrador, desarrollador y vendedor.

- **Administrador:** Posee todos los privilegios del sistema tanto para crear, modificar, borrar perfiles de usuarios, realizar el mantenimiento de la base de datos, etc.

- **Desarrollador:** Encargado de implementar un modelo mediante un lenguaje de programación, el mismo que una vez copilado puede entender cualquier dispositivo Android.

- **Vendedor:** Consultar catálogos, productos, promociones, cartera del cliente, registrar pedidos además de emitir un registro de visita.

1.2.2.3 Autenticación

En el ambiente corporativo el acceso no autorizado es un gran riesgo para la empresa ya que existe tecnología suficiente para romper la seguridad a nivel de claves de acceso. La solución planteada es crear un mecanismo de autenticación basada en TOKEN, es decir generar claves aleatoriamente disminuyendo la probabilidad que un usuario malintencionado utilice dicha clave.

Al implementar un mecanismo de autenticación fuerte, el servidor es quien elige una clave y sirve como base de datos de todas las contraseñas, a su vez esta clave es transmitida al dispositivo móvil donde se queda almacenado. Por lo que solo el usuario puede generar la clave y sólo el servidor decir si es un usuario legítimo [27]. A continuación los diferentes tipos de token existentes.

	**Token Mobile** Token para teléfonos móviles se encuentran en una variedad de equipos: BlackBerry, Android, iPhone, Windows Mobile, etc.
	**Token Web** Permite la autenticación fuerte sin la necesidad de distribuir un software a sus usuarios finales. De una forma sencilla los usuarios pueden registrar varias máquinas y luego utilizar cualquiera de los navegadores Web.
	**Token para PC** Puede ser distribuido de forma fácil en un Website público o instalado como parte de una configuración de fábrica en una máquina por un administrador.

Tabla 2. Tipos de Token [34].

1.2.2.4 Seguridad

Una aplicación móvil puede ser accedida desde diferentes tipos de dispositivos, es indispensable realizar un análisis de riesgos e identificar las posibles amenazas. En la siguiente tabla expondremos los posibles incidentes y estrategias para minimizar estos riesgos.

FACTOR RIESGO	ESTRATEGIA DE CONTROL
Debido al tamaño de los dispositivos móviles pueden ser robados o perdidos con mayor facilidad, existiendo el riesgo de acceso a la información corporativa por parte de personas no autorizadas.	▪ Mecanismos de autenticación como filtro inicial antes de poder acceder al dispositivo y al contenido almacenado. ▪ Cifrado a nivel de servidor y en el terminal, estableciendo una autenticación mutua.
Uso de redes no confiables (internet), la transmisión de información corporativa puede ser accedido por terceros.	▪ Encriptación de claves a nivel de servidor para que solo el administrador maneje el acceso.
Perdida de confidencialidad al conectar un dispositivo móvil con un ordenador.	▪ Registro previo de cada dispositivo móvil en el que se instalará la aplicación.

Tabla 3. Riesgos y estrategias al momento de crear aplicaciones móviles [25].

1.2.3 Comunicación

Para la comunicación es importante definir algunas reglas que debe seguir un programa para utilizar los servicios a nivel de capa de transporte TCP/IP.
Un socket es el punto final de una comunicación bidireccional entre dos programas que intercambian información a través de internet [27]. Existen dos tipos de sockets: los sockets stream y los sockets datagram.

1.2.3.1 Sockets stream (TCP)

Basado en el protocolo TCP, ofrece un servicio orientado a la conexión es decir antes de transmitir información se debe establecer una conexión tanto del lado del servidor como del cliente. Una vez que los dos sockets estén conectados podremos transmitir los datos en ambas direcciones.

1.2.3.2 Sockets Datagram (UDP)

Basado en el protocolo UDP, ofrece un servicio no orientado a la conexión, es decir podremos enviar información sin la necesidad de haber establecido una conexión previa.

La gran ventaja que aporta es que apenas introduce sobrecarga a la información transmitida, sin embargo no se garantiza que el datagrama llegue duplicado, perdido o llegue en un orden diferente [1].

1.2.4 Sincronización en el área de ventas

El escenario tecnológico que utilizará AGROTA CIA.LTDA está basado en la sincronización bidireccional, con las siguientes premisas tecnológicas:

- ✓ La sincronización permitirá asegurar que los datos existentes en el servidor serán los mismos que se encuentren en el móvil.
- ✓ La primera sincronización será posible realizarla desde redes locales, conexiones inalámbricas o accesos a través de internet, luego se podrá continuar trabajar localmente.
- ✓ La sincronización promoverá la escalabilidad del negocio, permitiendo realizar transacciones cuando varios empleados estén conectados.

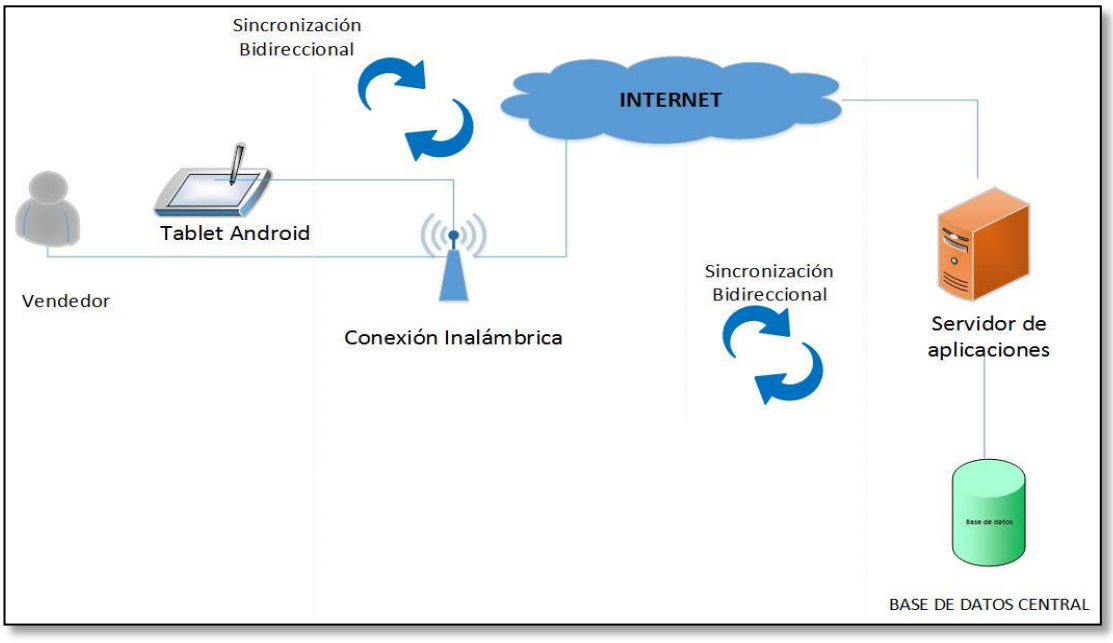


Ilustración 10. Esquema de sincronización AGROTA Cía. Ltda. [El autor].

Sin embargo el gran problema de la sincronización es transmitir solo los cambios en el estado del sistema y no el estado completo, de esta forma nos evitaremos enviar información que es innecesaria. A continuación se presentan algunas ventajas al usar este esquema:

- Optimizar las operaciones y consultas de la aplicación.
- Optimizar los recursos de los dispositivos móviles.
- Actualizar solo la información diaria de productos más vendidos.
- Enviar únicamente los cambios generados y no toda la información.

1.2.4.1 Mecanismos de sincronización en el área de ventas

Los eventos de negocio nos permiten representar solo los cambios en el estado del modelo y no todo el esquema de datos. Tenemos dos paradigmas los eventos push y pull.

- ***Eventos push***

La tecnología push se implementa en el lado del servidor permitiendo mantener una comunicación abierta de tal forma que tan pronto el servidor reciba un evento pueda enviar la información correspondiente al cliente [13].

Esta información puede enviarse en formato HTML, JSON o XML. Entre algunas de las tecnologías que hacen uso de los eventos push tenemos:

- ✓ **Cloud to Device Messaging (C2DM) de Google:** Se encarga de la gestión de cola y envío de las notificaciones a al cliente. De la misma manera, las aplicaciones no tienen que estar ejecutándose en el momento de recibir la notificación para reactivarse e interactuar con el servidor [13].

- ✓ **Push mail de BlackBerry:** Es el primer servicio que permite comunicaciones push para correo electrónico en movilidad.

- ✓ **Google Cloud Messaging para Android:** Es un servicio que permite enviar los datos de un servidor a un dispositivo móvil, así como recibir mensajes de otros dispositivos que estén en la misma conexión [14].

- ***Eventos pull***

La petición se realiza en el lado del cliente, si el factor comunicación en tiempo real no es indispensable (offline) podríamos plantear utilizar mecanismos tipo PULL. Es decir, nuestra aplicación lanzaría una consulta periódica al servidor en busca de novedades o mensajes especialmente dirigidos al usuario [14].

Entre algunas de las tecnologías que hacen uso de los eventos pull tenemos:

- ✓ **PullToRefresh:** Dispara un evento de actualización el cual nos permite ver las notificaciones existentes.

- ✓ **Gmail en Android:** Envían nuevos correos electrónicos a los clientes a medida que entran al servidor de correo.

- ✓ **WordPress para Android:** Realiza una actualización del contenido de una forma automática.

1.2.5 Servicios Web

"*El consorcio W3C* [1] *define los Servicios Web como sistemas de software diseñados para soportar una interacción interoperable máquina a máquina sobre una red* " [23]. Se trata de un API que es publicado, localizado e invocado a través de la web.

1.2.5.1 Alternativas en el servicio web

Existen tres enfoques diferentes al momento de definir un servicio web, es lo que se le conoce como arquitectura del servicio web [15].

- **Llamadas a Procedimientos Remotos (RPC):** Mecanismo que permite que los programas se ejecuten desde una máquina diferente.

[1] W3C - World Wide Web Consortium (Red Mundial Global)

- **Arquitectura Orientada a Servicios (SOA):** Es un protocolo orientado a mensajes, cada mensaje sigue estrictamente la sintaxis expresada en XML.

- **Transferencia de Estado Representacional (REST):** Una arquitectura basada en la solicitud de recursos, utiliza directamente el protocolo HTTP, por medio de las operaciones GET, POST, PUT y DELETE [15].

1.2.5.2 Servicios Web basados en SOAP

El protocolo de acceso simple (SOAP) es un protocolo comúnmente utilizado en el ámbito corporativo ya que la mayoría de empresas tienen sus sistemas basados en .NET o Java.

Considerando que la aplicación móvil debe integrarse con el sistema actual de la empresa basado en Visual Studio .Net usaremos SOAP por la facilidad para crear métodos y objetos de negocio.

Dado que Android no incluye ningún tipo de soporte para el acceso a servicios web SOAP, utilizaremos la librería ksoap2-android. Una librería ligera y eficiente que permite usar de una forma fácil los servicios web que utilicen SOAP [16].

1.2.5.3 Servicios Web basados en REST

Una arquitectura para el diseño de servicios web basada en los siguientes principios:

- **Arquitectura cliente servidor:** Define dos agentes: cliente, servidor y la interfaz de comunicación para separar responsabilidades.

- **Sin estado:** Son servicios que no mantienen un estado asociado a un cliente, cada petición realizada es independiente de la siguiente [1].

- **Caché:** Implica el uso de la caché para optimizar solicitudes, es decir una vez realizada la primera petición el resto pueda apoyarse en la caché [1].

- **Arquitectura en capas:** Su arquitectura modular promueve la escalabilidad y adaptación hacia nuevas tecnologías.

1.3 Aplicaciones móviles en el área de ventas

El ciclo de venta de un determinado producto conlleva una serie de procesos tediosos, mediante la implementación de varias herramientas dinámicas lograremos brindar un servicio más personalizado al cliente, agilizar el proceso de emisión de pedidos y retroalimentar a la empresa sobre los productos más vendidos, cobros realizados, etc. Algunas herramientas que nos ayudarán a alcanzar los objetivos son:

1.3.1 Catálogos Electrónicos

Un catálogo electrónico es una aplicación online, es decir una interfaz gráfica generalmente una página en la que se muestran los productos y servicios ofrecidos por la empresa. Dentro del comercio electrónico se distinguen las operaciones realizadas entre empresas y consumidores, y aquellas realizadas entre empresas [17]. En este sentido, los catálogos electrónicos son una excelente forma de comunicación y marketing. Existen diferentes tipos de catálogos según el modo en que pueden aparecer en internet:

- **Catálogo Electrónico tipo FlipPage (Tipo libro electrónico):** Simula un efecto natural al cambiar de página, como si se tuviera un documento físico, son muy utilizados por usuarios que disponen pantallas táctiles. Los catálogos "Page Flip" o "Flip Page" aprovechan las funcionalidades de un documento tradicional electrónico ofrece: búsqueda a lo largo del documento, saltar a cierta página, enlaces a websites externos, integración de animaciones, integraciones multimedia, video y audio [18].

Ilustración 11. Catálogo FlipPage [18].

- **Catálogos Electrónicos con opción de carrito de compra:** Consiste en crear un catálogo categorizado en el cual existe la opción de carrito de compra online**.**

- **Catálogos Electrónicos Autoadministrables:** Es un catálogo dinámico en el cual la información se actualiza automáticamente además de permitir al usuario administrar su negocio independientemente de su ubicación.

1.3.2 Geolocalización en Android

La Geolocalización se define como la utilización de un *"sistema de coordenadas para representar de forma gráfica y entendible la posición de un objeto en el espacio, de forma que puede reconocerse su ubicación*" [36].

Android integra varios servicios de localización como: manejo de mapas, precisión de ubicación, opciones de seguridad, búsquedas inteligentes, administración de rutas, etc.

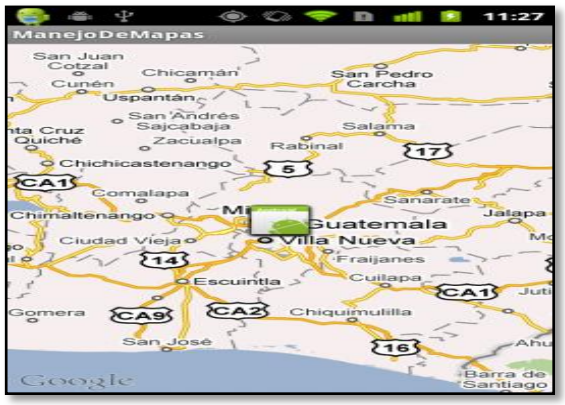


Ilustración 12. Manejo de mapas en Android [19].

1.3.3 Facturación Electrónica

Automatiza la emisión de facturas tradicionales, haciendo el proceso más simple, rápido y barato permitiéndonos dejar atrás los errores provocados por el factor humano. La facturación electrónica crea un nuevo modelo de negocio mejorando la precisión de la información, seguridad, reducción de tiempos en procesos administrativos además de evitar el deterioro de documentos.

1.3.4 Aplicaciones de Agenda Electrónica

La agenda electrónica permite organizar contactos, notas, recordatorios, los mismos pueden ser integrados a diferentes dispositivos móviles con la finalidad de proporcionar los siguientes beneficios:

- Personalización
- Accesibilidad
- Organización de tareas.
- Sincronización de calendario con otros dispositivos móviles.
- Uso de aplicaciones personales como: libretas, direcciones, agenda, etc.

1.3.5 Tiendas Virtuales

Si bien la accesibilidad a una tienda física no es siempre fácil para el usuario, mediante una tienda virtual el cliente puede acceder a una página web para consultar o adquirir productos que sean de su interés de una forma rápida y desde cualquier lugar [31].

Esta herramienta ha permitido incorporar una especificación virtual completa de las líneas de productos incluyendo una descripción detallada del precio, especificaciones, imágenes, ubicación, calificación del producto, productos más vendidos, etc. También han obligado al desarrollador a perfeccionar los estilos de diseño para conseguir aplicaciones que sean fáciles de usar y sobre todo que tengan coherencia visual.

1.4 Aplicación móvil off-line

1.4.1 Lenguajes de programación más usados a nivel empresarial

A nivel empresarial los dispositivos inteligentes han permitido que la información que permanecía estática en las base de datos, esté presente en cualquier lugar donde se realicen transacciones de negocio. En la actualidad existen muchas herramientas que ayudan a promocionar los productos tales como catálogos electrónicos, combos promocionales, publicidad en línea, etc.

Sin embargo el desarrollo de una aplicación móvil depende de varios factores tales como: el lenguaje de programación, sistema operativo, capacidad de almacenamiento, interfaces, comunicación, sincronización, servicios, etc.

De acuerdo al Ranking de Lenguajes de Programación (TIBOE), las primeras posiciones están reservadas por C, Java y Objetive C, como podemos observar la tendencia actual está enfocada hacia la creación de aplicaciones móviles. Así Android (Java) y IOS (Objetive-C) lideran los primeros lugares a nivel mundial [11]. Ilustración 13.

Nov 2014	Nov 2013	Change	Programming Language	Ratings	Change
1	1		C	17.469%	-0.69%
2	2		Java	14.391%	-2.13%
3	3		Objective-C	9.063%	-0.34%
4	4		C++	6.098%	-2.27%
5	5		C#	4.985%	-1.04%
6	6		PHP	3.043%	-2.34%
7	8	^	Python	2.589%	-0.52%
8	10	^	JavaScript	2.088%	+0.04%
9	12	^	Perl	2.073%	+0.55%
10	11	^	Visual Basic .NET	2.061%	+0.09%
11	-	^^	Visual Basic	1.657%	+1.66%
12	31	^^	R	1.548%	+1.14%
13	9	vv	Transact-SQL	1.408%	-1.11%
14	13	v	Ruby	1.211%	-0.09%
15	17	^	Delphi/Object Pascal	0.957%	+0.31%
16	23	^^	F#	0.892%	+0.39%
17	18	^	PL/SQL	0.870%	+0.27%
18	-	^^	Swift	0.834%	+0.83%

Ilustración 13.Índice TIOBE de lenguajes de programación más utilizados hasta Noviembre del 2014 [11].

La Ilustración 14 refleja la gran aceptación de Java por parte de programadores y desarrolladores. Un buen programador va más allá del S.O, se define en la limpieza que tenga el código que crea, la mantenibilidad del mismo, la documentación, manejo de estándares, eficiencia, portabilidad, seguridad, son las primicias que han hecho de Java un lenguaje de programación universal y un competidor difícil de reemplazar.

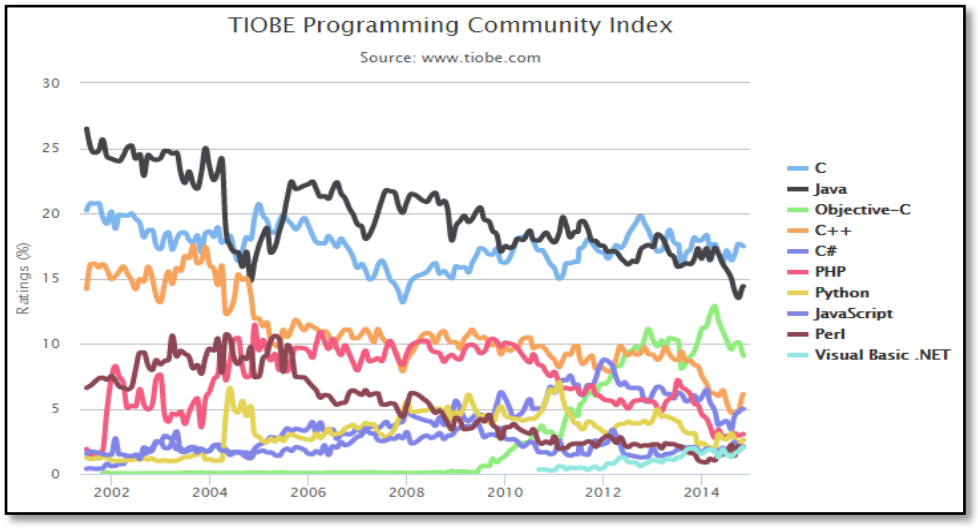


Ilustración 14. Ranking de lenguajes de programación (Noviembre 2014) [11].

1.4.2 Sistemas operativos de mayor demanda en los negocios

Desde el punto de vista operativo IOS era el sistema operativo por excelencia hasta el 2013. Hoy en día el panorama parece cambiar ya que Android ocupan una considerable parte del mercado mundial.

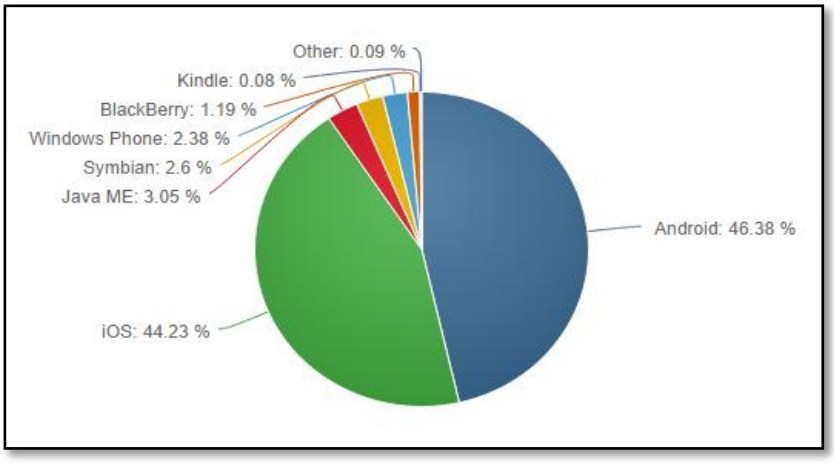


Ilustración 15. Estadísticas de Tecnologías Móviles (Octubre 2014) [12].

La tendencia actual no es muy positiva para IOS ya que ha perdido casi el 10% de terreno, pasando del 53,54% en marzo del 2013 a un 44,23% en octubre 2014. En cambio Android ha ganado parte del terreno perdido por IOS, pasando de un 19,27% en marzo del 2013 a un 46,38% en octubre del 2014. [12]. Actualmente los usuarios prefieren aplicaciones gratuitas, integración de servicios, facilidad de personalización, interfaz amigable, diversidad de dispositivos. Por lo que podemos decir que el mercado se muestra muy optimista para las aplicaciones creadas en Android.

1.4.3 Estrategias y herramientas para crear aplicaciones offline

La capacidad de utilizar aplicaciones móviles cuando no se está conectado a internet se ha convertido en una necesidad ya que por diversos factores tales como: ubicación física, condiciones climáticas, movilidad, no se puede garantizar una conectividad al 100%. Es por esta razón que se ha planteado varias estrategias y herramientas que nos ayudarán asegurar una buena experiencia off-line.

- ***Caché de aplicación:*** Consiste en utilizar la caché para guardar los recursos de la aplicación. Una aplicación móvil que funcione mediante la caché proporcionará las siguientes ventajas:

  - ✓ **Navegación fuera de línea:** Facilidad para registrar pedidos, consultar cartera del cliente, sin la necesidad de disponer una conexión a internet.

  - ✓ **Velocidad:** Los recursos al estar almacenados en la caché se cargan más rápido.

  - ✓ **Reducción de la carga del servidor:** El navegador solo descarga los recursos que han cambiado desde el servidor.

- ***Comprobación de la conectividad:*** En una aplicación offline es importante conocer cuando el usuario vuelve a conectarse para poder establecer una nueva sincronización con el servidor y cuando el usuario está en línea para mejorar la administración de la cola de peticiones en el servidor.

- ***Almacenamiento Local:*** El Almacenamiento Document Object Model (DOM) permite almacenar gran cantidad de datos de forma persistente y segura. Al hacer uso del método sessionStorage podemos guardar datos de forma persistente, mientras que con el LocalStorage los datos se guardaran mientras dure la sesión [21,22].

1.4.4 Aplicaciones Android off-line

- **Diccionario offline:** Aplicaciones gratuitas que soportan una variedad de idiomas, permitiendo que el usuario disponga de un diccionario multilingüe.

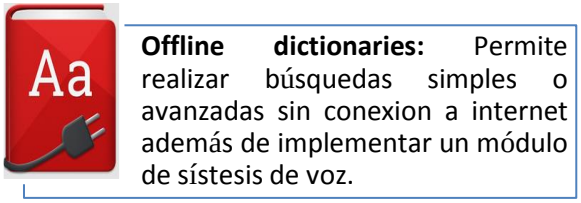


Ilustración 16. Ejemplo de aplicación diccionario offline para Android [20].

- **Google maps:** Ideal cuando se desea viajar, gracias al GPS offline podemos descargar una gran variedad de mapas y almacenarlos en memoria.

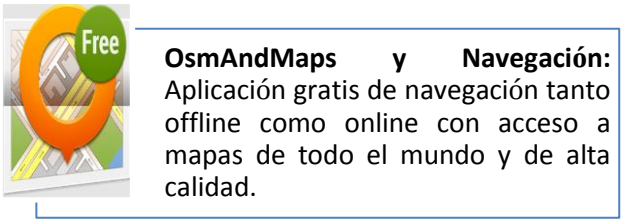


Ilustración 17. Ejemplo de aplicación GPS offline para Android [20].

- **Reproductores de música offline:** Permite buscar y escuchar canciones offline, además de crear listas de reproducción, compartir música y acceder a discografía.

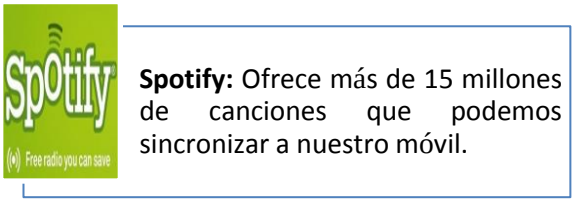


Ilustración 18. Ejemplo de aplicación música offline para Android [20].

CAPÍTULO 2

2. ANÁLISIS E IDENTIFICACIÓN DE LOS REQUERIMIENTOS DE LA EMPRESA

Introducción

La empresa Agrota Cía. Ltda. debido a la alta demanda de clientes, se encuentra en la necesidad de construir un sistema enfocado en el registro de pedidos. Mediante el documento de Especificación de Requerimientos de Software (SRS) se capturará la información necesaria para ayudar a los desarrolladores a entender las expectativas del cliente, así mismo este capítulo contiene descripciones útiles sobre los requerimientos funcionales, requerimientos no funcionales, stakeolders, interfaces de aplicación, módulos de trabajo, etc.

El siguiente documento ha sido elaborado según la Norma ISO/IEC/IEEE 29148:2011 para la SRS, presentando una descripción detallada del funcionamiento del sistema además de permitirnos crear un documento soporte para el desarrollador y plantear una línea base para el diseño e implementación.

Propósito

El siguiente capítulo describe la ESPECIFICACION DE REQUERIMIENTOS DEL SOFTWARE PARA LA TOMA DE PEDIDOS. Este documento contempla los lineamientos que se deben seguir para definir una especificación global de la aplicación, documentar las restricciones de los interesados, definir los servicios a ofrecer y sobre todo establecer los límites del proyecto.

Mediante la especificación de requerimientos el cliente está obligado a analizar cuidadosamente las necesidades a cubrir ya que en el caso de existir algún cambio futuro, los desarrolladores contarán con un documento contractual que valide el convenio emitido.

2.1 Identificación de los Requerimientos

"Todas las acciones que hacen falta para que un usuario consiga su objetivo se traducen en requerimientos" [37]. Sin embargo es importante establecer prioridades para decidir si ese requerimiento merece ser incluido, para no terminar con un producto saturado de funciones que nadie use y arruine la experiencia del usuario.

Con una buena identificación de necesidades podremos manejar proyectos complejos y mejorar la calidad de nuestros sistemas. El producto final complementará el valor del usuario en diferentes contextos o etapas del proyecto con la funcionalidad esperada [35].

2.1.1 Interfaces internas

Uno de los aspectos más importantes de la ingeniería de requerimientos es la comunicación, característica que vuelve el proceso complejo por la alta presencia del factor humano [39].

Las interfaces internas por su parte facilitan la interacción dispositivo con dispositivo, es decir emplean dispositivos de hardware para proteger la información. Estos dispositivos pueden ir desde pequeños disco magnéticos hasta grandes servidores que están al servicio de varios clientes.

2.1.1.1 Servidor Central

El servidor central de la empresa contiene la información sobre los productos, promociones, descuentos, combos, etc. los mismos que mediante un mecanismo de sincronización podrán ser accedidos por el vendedor al momento de registrar una orden de pedido offline.

2.1.2 Interfaces externas

Las interfaces externas permiten describir de forma detallada la interacción con otros sistemas así como los requisitos de conexión, librerías necesarias para manipular la base de datos móvil SQLite.

Un principio clave de diseño consiste en separar los datos, la lógica y la presentación, es importante considerar que pequeños cambios pueden repercutir en complicadas modificaciones de código, es por esta razón que el diseño arquitectónico representa un punto clave para la integración de ambos sistemas.

Debido a que las interfaces representan la primera experiencia del usuario, su diseño y construcción será detallada con mayor profundidad en el capítulo 4.

2.1.2.1 Base de datos SQLite

La interfaz entre la aplicación móvil y el hardware lo proporciona la librería SQLite, mediante esta librería podremos manejar de una forma ágil todas las operaciones SQL como índices, triggers, vistas, múltiples tablas, etc.

Su diseño ha sido orientado hacia el desarrollo de aplicaciones personales y livianas ya que los datos son almacenados en un único archivo de texto plano. Esto hace que cada usuario pueda crear tantas bases de datos como desee sin la necesidad de la intervención de un administrador que gestione los espacios de trabajo, usuarios, permisos de acceso, etc. [38].

SQLite a nivel empresarial representa la solución de movilidad permitiendo disponer de información al instante, responder rápidamente a las necesidades del negocio sin depender de una localización física.

El vendedor contará con un dispositivo inteligente que le permitirá organizar y agilizar su trabajo operativo además de mejorar la comunicación con sus clientes con la facilidad de llevar en un dispositivo portable toda la información necesaria para su desempeño laboral.

2.1.3 Funciones

Requerimientos No Funcionales					
Rendimiento	**Seguridad**	**Disponibilidad**	**Comunicación**	**Mantenibilidad**	**Portabilidad**
-El tiempo de respuesta para cada solicitud será inmediato. -La aplicación soportará dos o más usuarios al mismo tiempo ofreciendo un manejo eficiente del sistema. -La aplicación móvil permitirá la actualización de la información agregada o modificada en el servidor central. -El vendedor podrá realizar consultas ágiles mediante filtros por fechas sin la necesidad de acceder a toda la base de datos móvil.	-La seguridad de la aplicación se controlará mediante un sistema de autenticación basado en un usuario y contraseña. -También se Implementará encriptación a nivel de claves de acceso de esta forma la información podrá ser leída solo por personas autorizadas.	-Los pedidos podrán ser registrados de forma offline. -En el caso que el administrador cambie la clave de acceso, el vendedor podrá acceder mediante la clave original con la que realizo la primera autenticación al estar conectado a internet.	-La aplicación móvil soportará la arquitectura 3 capas. -La aplicación móvil emitirá notificaciones. -La aplicación WISE emitirá mensajes para controlar la tecla *back* al momento de navegar entre las diferentes pantallas.	-El sistema incluirá un manual usuario y manual de administrador. -La documentación de ayuda contendrá la actualización de software y hardware recomendable. -La aplicación móvil será creada de tal forma que en un futuro se pueda agregar más módulos si la empresa lo requiere.	-Utilizar el lenguaje y plataforma JAVA. -Utilizar como base de datos móvil SQLite. -Utilizar como servidor de base de datos central. MySQL. -Utilizar componentes, herramientas y librerías Android

Tabla 4. Especificación de Requerimientos No Funcionales. **Fuente:** El autor.

Requerimientos Funcionales	
RF – 001	El sistema implementará un sistema de autenticación basado en el ingreso de usuario y contraseña.
RF – 002	El sistema permitirá actualizar tanto los datos nuevos como los modificados en el servidor central.
RF – 003	El sistema permitirá listar la información del cliente (nombre, apellido, teléfono, dirección, ciudad, nombreComercial, email, calificación).
RF – 004	El sistema permitirá registrar una orden de pedido offline considerando (cliente, descuento, producto, forma de pago, valor de transporte y observaciones).
RF – 005	El sistema permitirá listar productos categorizados considerando (nombreProducto, código, PVP, descuento, precioCliente).
RF – 006	El sistema permitirá agregar productos por catálogo directamente a la orden de pedido considerando (cliente, nombreProducto, descuento).
RF –007	El sistema permitirá listar todas las órdenes de pedido emitidas por el vendedor.
RF –008	El sistema permitirá listar la cartera del cliente considerando (número de factura, fechaEmisión, valor, retención, saldo, forma de pago y cuotas).
RF –009	El sistema permitirá actualizar la información que incluye la cartera del cliente.
RF –010	El sistema permite registrar los cobros efectuados al cliente, emitir un registro de visita y registro de clientes nuevos.

Tabla 5. Identificación de Requerimientos Funcionales. **Fuente:** El autor.

2.1.3.1 Especificación de Requerimientos Funcionales

RF - 001	**Ingresar al sistema**	**Versión: 1.1**
Autores:	Johanna Picón, Sebastián Zhinin	
Fuentes:	Documento de Elicitación	
Descripción:	Realiza la autentificación de usuarios mediante un sistema de logueo.	
Actor:	Vendedor-Administrador	
Entradas:	Ingresar a la aplicación WISE	
Proceso:	Paso	Acción
	1)	Ingresar usuario y contraseña.
	2)	El sistema verifica los datos para proporcionar el permiso de acceso.
	3)	El usuario termina la sesión.
Salidas:	Guardar datos en la base de datos móvil.	
Pos condiciones:	Paso	Acción
	1)	Validar información con el sistema central, si los datos son incorrectos emitir un mensaje indicando el inconveniente.
Prioridad:	ALTA	MEDIA BAJA

Tabla 6.RF-001**:** Ingresar al sistema. **Fuente:** El autor.

RF - 002	Sincronizar Datos Servidor Central		Versión: 1.1
Autores:	Johanna Picón, Sebastián Zhinin		
Fuentes:	Documento de Elicitación		
Descripción:	Permite descargar todos los datos necesarios del servidor central.		
Actor:	Vendedor		
Entradas:	Ingresar al sistema		
Proceso:	Paso	Acción	
	1)	El vendedor visualiza el inicio de descarga de datos.	
	2)	El dispositivo móvil dispone de toda la información necesaria para funcionar.	
Salidas:			
Pos condiciones	Paso	Acción	
	1)	Este proceso se llevar a cabo para realizar la primera sincronización, luego se puede trabajar localmente.	
Prioridad:	ALTA	MEDIA	BAJA

Tabla 7.RF-002: Sincronizar Datos Servidor Central. **Fuente:** El autor.

RF - 003	Listar Cliente		Versión: 1.1
Autores:	Johanna Picón, Sebastián Zhinin		
Fuentes:	Documento de Elicitación		
Descripción:	Permite listar la información del cliente.		
Actor:	Vendedor		
Entradas:	Ingresar al sistema		
Proceso:	Paso	Acción	
	1)	El vendedor ingresa al módulo cliente.	
	2)	El vendedor visualiza la información de detalle del cliente.	
Salidas:			
Pos condiciones	Paso	Acción	
	1)	El sistema emitirá una lista que incluirá toda la información del cliente, permitiendo al vendedor identificar e ubicar al cliente.	
Prioridad:	ALTA	MEDIA	BAJA

Tabla 8.RF-002: Listar cliente. **Fuente:** El autor.

RF – 004	Registrar Orden de Pedido	Versión: 1.1
Autores:	Johanna Picón, Sebastián Zhinin	
Fuentes:	Documento de Elicitación	
Descripción:	Permite registrar una orden de pedido independientemente de contar con una conexión a internet.	
Actor:	Vendedor	
Entradas:	Ingresar al sistema.	
Proceso:	Paso	Acción
	1)	El vendedor selecciona un cliente, el mismo que tendrá una calificación asociada para poder aplicar el descuento.
	2)	El vendedor ingresa al módulo orden de pedido y escoge los productos.
	3)	Procede a elegir la forma de pago (crédito o prepago).
	4)	Ingresar el valor de transporte negociado con el cliente.
	5)	Escribir las observaciones de la orden de pedido.
Salidas:	Orden de pedido guardado en la base de datos	
Pos condiciones:	Paso	Acción
	1)	Validar la información, si el registro no se ha guardado satisfactoriamente el sistema emitirá un mensaje de error.
Prioridad:	ALTA	MEDIA BAJA

Tabla 9.RF-003: Registrar orden de pedido. **Fuente:** El autor.

RF – 005	Listar Productos	Versión: 1.1
Autores:	Johanna Picón, Sebastián Zhinin	
Fuentes:	Documento de Elicitación	
Descripción:	Permite listar productos de acuerdo a la categorización ya sea por: líneas, categorías o marcas.	
Actor:	Vendedor	
Entradas:	Ingresar al sistema	
Proceso:	Paso	Acción
	1)	El vendedor ingresa al módulo producto
	2)	El vendedor busca el producto dentro de una categorización.
Salidas:		
Pos condiciones:	Paso	Acción
	1)	El sistema emitirá una lista con todos los productos pertenecientes a la categorización.
Prioridad:	ALTA	MEDIA BAJA

Tabla 10.RF-004: Listar productos. **Fuente:** El autor.

RF -006	Agregar productos por catálogo a la orden de pedido	Versión: 1.1
Autores:	Johanna Picón, Sebastián Zhinin	
Fuentes:	Documento de Elicitación.	
Descripción:	Permite agregar productos por catálogo a la orden de pedido.	
Actor:	Vendedor	
Entradas:	Ingresar al sistema.	
Proceso:	Paso	Acción
	1)	El vendedor accede al módulo producto y verifica la existencia del mismo.
	2)	Seleccionar la opción agregar producto y digitar la cantidad deseada.
Salidas:	Producto guardado en la base de datos	
Pos condiciones:	Paso	Acción
	1)	Se podrá escoger solo un ítem de cada grupo del combo, caso contrario se emitirá un mensaje de error.
Prioridad:	ALTA	MEDIA BAJA

Tabla 11.RF-005: Agregar productos por catálogo a la orden de pedido. **Fuente:** El autor.

RF - 007	Listar orden de pedido	Versión: 1.1
Autores:	Johanna Picón, Sebastián Zhinin	
Fuentes:	Documento de Elicitación	
Descripción:	Permite visualizar todas las órdenes emitidas.	
Actor:	Vendedor	
Entradas:	Ingresar al sistema	
Proceso:	Paso	Acción
	1)	El vendedor ingresa al módulo de orden de pedido.
	2)	Escoger la opción *Ver Ordenes*
	3)	Filtrar el pedido de acuerdo a una fecha o por cliente.
Salidas:		
Pos condiciones:	Paso	Acción
	1)	El sistema emitirá una lista con todos los pedidos creados. Si se desea agilizar la búsqueda se puede utilizar un filtro por fecha o por cliente.
Prioridad:	ALTA	MEDIA BAJA

Tabla 12.RF-007: Listar orden de pedido. **Fuente:** El autor.

RF -008	Lista cartera del cliente	Versión: 1.1
Autores:	Johanna Picón, Sebastián Zhinin	
Fuentes:	Documento de Elicitación	
Descripción:	Permite visualizar toda la información de detalle que incluye la cartera del cliente.	
Actor:	Vendedor	
Entradas:	Ingresar al sistema	
Proceso:	Paso	Acción
	1)	Seleccionar un cliente para ver el detalle de la cartera.
	2)	El vendedor ingresa al módulo cartera cliente.
	3)	El vendedor visualiza todas las facturas emitidas al cliente.
	4)	Aplicar un filtro por fecha para agilizar la búsqueda.
Salidas:		
Pos condiciones:	Paso	Acción
	1)	La aplicación emitirá una lista con el detalle de la forma de pago y cuotas establecidas.
Prioridad:	**ALTA** ■	☐ **MEDIA** ☐ **BAJA**

Tabla 13.RF-008: Listar cartera del cliente. **Fuente:** El autor.

RF -009	Actualizar cartera del cliente	Versión: 1.1
Autores:	Johanna Picón, Sebastián Zhinin	
Fuentes:	Documento de Elicitación	
Descripción:	Permite actualizar la cartera de un cliente específico, esto incluye: facturas, cuotas y forma de pago.	
Actor:	Vendedor	
Entradas:	Ingresar al sistema	
Proceso:	Paso	Acción
	1)	Seleccionar un cliente
	2)	Ingresa al módulo cartera cliente.
	3)	El vendedor pulsa el botón actualizar cartera.
Salidas:		
Pos condiciones:	Paso	Acción
	1)	El sistema emitirá un mensaje indicando la actualización exitosa, caso contrario se mostrara un mensaje de error.
Prioridad:	**ALTA** ■	☐ **MEDIA** ☐ **BAJA**

Tabla 14. RF-009: Actualizar cartera del cliente. **Fuente:** El autor.

RF -010	Registrar Hoja de Visita	Versión: 1.1
Autores:	Johanna Picón, Sebastián Zhinin	
Fuentes:	Documento de Elicitación	
Descripción:	Permite llevar un registro de los cobros de las facturas del cliente, a la vez se puede ingresar datos de los nuevos clientes y registrar el estado de la visita.	
Actor:	Vendedor	
Entradas:	Ingresar al sistema	
Proceso:	Paso	Acción
	1)	Seleccionar un cliente
	2)	Ingresa al módulo hoja de visita
	3)	El vendedor ingresa al menú de hoja de visita. Aquí se exponen tres opciones: registro de visita, clientes nuevos y cobros.
Salidas:	Registro guardado en la base de datos.	
Pos condiciones:	Paso	Acción
	1)	El sistema emitirá un mensaje indicando que el registro se almaceno correctamente.
Prioridad:	**ALTA** ■	☐ **MEDIA** ☐ **BAJA**

Tabla 15.RF-010: Registrar Hoja de Visita. **Fuente:** El autor.

2.1.3.2 Requerimientos de Usabilidad

"La usabilidad se define como el grado en el cual un producto puede ser utilizado por sus usuarios para lograr metas con efectividad, eficiencia y satisfacción en un determinado contexto de uso" [40]. Los requerimientos de usabilidad que persigue la aplicación móvil son:

- ✓ Proporcionar una interfaz amigable para el vendedor, cada módulo contará con un icono distintivo que permita entender la funcionalidad del mismo.

- ✓ La aplicación móvil será desarrollada bajo el concepto de software libre utilizando Android como plataforma de desarrollo para facilitar la integración de nuevas tecnologías.

- ✓ La aplicación WISE será portable ya que podrá ser instalada en cada uno de los dispositivos móviles de los vendedores sin presentar diferencias en cuanto al nivel de performance.
- ✓ La aplicación WISE garantizará una respuesta rápida a las solicitudes de los vendedores.
- ✓ La aplicación móvil contendrá la información necesaria para detallar la funcionalidad de cada módulo, además de proporcionar mensajes de ayuda, advertencias y errores.

2.1.3.3 Requerimientos de Rendimiento

- ✓ La aplicación tendrá que presentar los más altos índices de rendimiento, para esto se implementará algoritmos dinámicos que aceleren el proceso de búsqueda de clientes, productos, registro de pedidos, etc.

- ✓ El sistema soportará las solicitudes de dos o más usuarios concurrentes ofreciendo un nivel de performance óptimo.

- ✓ El sistema responderá a las solicitudes de cada uno de los vendedores en la brevedad posible.

- ✓ El sistema permitirá al vendedor consultar la información del estado del pedido y cartera de forma actualizada.

2.1.3.4 Requerimientos de Base de Datos Lógica

- ✓ El acceso a las capacidades de la base de datos del servidor central será accesible únicamente para el administrador del sistema o quien esté a cargo del mismo y el personal designado por la empresa Agrota Cía. Ltda.

- ✓ Las restricciones de integridad serán planteadas a nivel de base de datos móvil, mediante la implementación del cifrado de claves de acceso.

- ✓ Se definirá métodos de acceso y concesiones de autorización para permitir el ingreso a la base de datos móvil.

2.2 Análisis de la solución

2.2.1 Descripción General del Problema

Actualmente la empresa Agrota Cía. Ltda. realiza las órdenes de pedido de forma manual, el vendedor visita al cliente para tomar su orden y debe regresar a la central para poder emitirlo. Tomando en cuenta que el proceso es ineficiente y poco óptimo, la empresa se ha visto en la necesidad de crear un nuevo modelo de negocio para dejar atrás los documentos en papel y agilizar el trabajo diario del vendedor. La idea es crear una aplicación móvil que consolide múltiples servicios para automatizar el registro de pedidos offline, consultar catálogos, consultar combos, promocionales, etc.

2.2.2 Perspectiva del Producto

Realizar el análisis, diseño e implementación de una aplicación móvil encaminada hacia el ámbito empresarial, el mismo que será integrado al sistema central de la empresa. Aquí debemos considerar que debe existir una sincronización continua para garantizar que los procesos se ejecuten adecuadamente. A continuación se presenta un esquema de bloques que detalla el funcionamiento global de la integración de ambos sistemas.

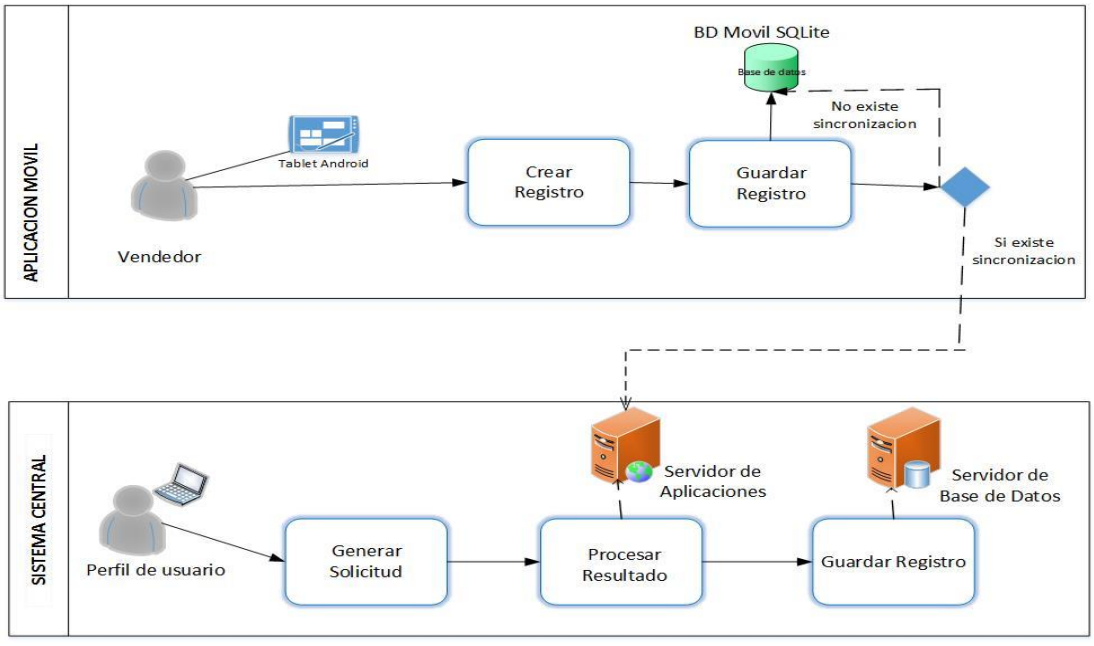


Ilustración 19.Diagrama de Bloques de la Integración de ambos sistemas .**Fuente:** El autor

La Ilustración 19 presenta un esquema del funcionamiento del sistema final. Por un lado tenemos el sistema móvil, encargado de crear un registro de pedido, almacenarlo en la base de datos del móvil y enviarlo al servidor central mediante un sistema de sincronización. Por otro lado tenemos al servidor central, conformado por un servidor de aplicaciones encargado de procesar dicha solicitud y almacenarlo en el servidor de base de datos central. Una vez que la persona delegada por la empresa autorice la orden podrá ser despachada.

2.2.2.1 Interfaces de usuario

La interfaz de usuario representa un conjunto de gráficos, guías, ventanas (formularios) con los que el usuario debe interactuar para realizar una operación determinada. Android nos presenta una serie de herramientas para manejar pantallas táctiles, eventos de usuario, permitiéndonos crear un entorno amigable e interactivo. Es importante también indicar que las interfaces de usuario comprenden un conjunto de ayudas que contribuyen a mejorar la experiencia del usuario. Estas ayudas se presentarán mediante una serie de formularios que incluyen:

- ✓ Formulario de logue
- ✓ Formulario del cliente
- ✓ Formulario de productos
- ✓ Formulario de registro de pedidos
- ✓ Formulario de consulta de cartera
- ✓ Formulario para la hoja de visita
- ✓ Formulario para verificar cuotas o abonos realizados por el cliente.
- ✓ Menús desplegables
- ✓ Catálogos de productos
- ✓ Botones
- ✓ Mensajes de errores.

2.2.2.2 Interfaces de software

La función que cumple las interfaces de software es entregar información sobre las peticiones realizadas al sistema. Comprende todo el conjunto de pantallas con las que el usuario deberá interactuar.

Plataforma de Desarrollo:	Java EE, Android
Lenguajes de Programación:	Java, XML, SQL.
Implementación vistas XML	Android
Servidor de Aplicaciones:	Internet Information Services(IIS) 7
Base datos Servidor Central	MySql
Base de Datos Móvil:	SQLite

Tabla 16. Requerimientos de interfaz del software. **Fuente:** El autor.

2.2.2.3 Interfaces de hardware

Las interfaces de hardware indican la forma en la que se va almacenar la información y cómo la aplicación móvil permitirá el ingreso, procesamiento o entrega de información a los vendedores.

Tablet:	Samsung Galaxy Tab 3 Lite T110
Sistema Operativo:	Android
Versión	Android Jelly Bean 4.3
Velocidad del procesador:	1.2 GHz
Memoria:	RAM 1 GB - ROM 8 GB
Espacio de disco mínimo:	8 GB

Tabla 17. Requerimientos de interfaz de hardware del dispositivo móvil. **Fuente:** El autor.

Servidor:	Servidor HP ProLiant DL380 Gen8
Sistema Operativo:	Novell SUSE
Procesador Mínimo:	Intel Xeon E5-2407 (4 nucleos,2.2 GHz,10 MB,80W)
Procesador Recomendado:	procesador Intel Xeon de núcleo cuádruple de la serie 5400 a un máximo de 3,33 GHz
Memoria Mínima:	16gb
Memoria Recomendada:	64gb
Espacio en Disco mínimo:	1TB
Espacio en Disco recomendado:	5TB

Tabla 18. Requerimientos de interfaz de hardware del servidor central. **Fuente:** El autor.

2.2.2.4 Interfaces de comunicación

Para sincronizar los datos del sistema central con la aplicación móvil, es necesario emplear un mecanismo de comunicación que permita vincular las funciones de ambos sistemas. Para esto se utilizará los servicios web basados en el protocolo SOAP obteniendo así:

- ✓ Facilidad para integrar diferentes servicios.
- ✓ Fomentar el uso de estándares y protocolos.
- ✓ Interoperabilidad entre distintas plataformas.

2.2.2.5 Requisitos de adaptación

- La aplicación funcionará con una arquitectura tres niveles basada en el concepto de capas.

- Se asume que la aplicación funciona conjuntamente con el sistema central de la empresa, sin embargo en el caso que exista algún problema con el servidor, los pedidos serán almacenados en el móvil, una vez que se vuelva a tener conectividad con el servidor dichos pedidos podrán ser procesados.

- La aplicación será implementada en la versión Android 4.x en adelante, los usuarios con versiones anteriores no podrán instalar todos los servicios de la aplicación ya que existen métodos, clases y estilos que están disponibles a partir de una versión determinada.

2.3 Especificación de los módulos de trabajo

La aplicación móvil tiene varias funciones las cuales se encuentran divididas en módulos facilitando la navegación y usabilidad por parte del usuario. Al ingresar a la aplicación se presenta el menú principal, el mismo que está compuesto por seis módulos, a continuación se detalla una previa de la funcionalidad de cada módulo.

2.3.1 Módulo Menú

Es el módulo principal encargado de proporcionar acceso a todas las funcionalidades de la aplicación. Ofrece al usuario una lista de opciones para búsqueda, ayuda y formularios. También incluye una serie de submódulos, los mismos que serán detallados a continuación:

2.3.1.1 Módulo Pedidos

Permite registrar, consultar y visualizar una orden de pedido con el siguiente detalle: código, cliente, nombreComercial, calificaciónCliente, producto, descripción, precioUnit, cantidad, subtotal, promoción, forma de pago, iva, transporte y total. Para visualizar todas las órdenes registradas por el vendedor, se podrá acceder a la opción Ver Órdenes y agilizar la búsqueda aplicando un filtro por fecha.

2.3.1.2 Módulo Cliente

El vendedor podrá consultar los datos del cliente estos incluyen: nombres, apellidos, nombreComercial, teléfono, email, ubicación. También incluye información sobre la calificación del cliente, dato útil para aplicar el respectivo descuento a la orden.

- **Calificación del cliente:** El cliente obtendrá una calificación (A, B, C, D), la misma que es estructurada de acuerdo al volumen, pago y potencial. En el caso de que el cliente no tenga calificación no tendrá derecho al descuento.

2.3.1.3 Módulo Cartera

Contiene información sobre el estado de cuenta del cliente y forma de pago. Dado que el vendedor puede elegir entre dos opciones de pago (crédito o prepago).El siguiente módulo ayudará a llevar un registro de las cuotas o abonos cancelados por el cliente, diferenciando con color rojo las cuotas aún pendientes y con color verde las cuotas canceladas a su totalidad.

2.3.1.4 Módulo Productos

Contiene un catálogo virtual organizado por líneas, categoría y marcas. Cada producto contendrá información detallada sobre: nombreProducto, código, PVP, descuento y precioCliente. El vendedor podrá registrar una orden de pedido ya sea escogiendo ítems normales, ítems agrupados combo o ítems en promoción. Los productos pertenecientes a un combo o extra tendrán un solo precio final por el contrario los ítems en promoción el precio dependerá de la cantidad de productos adquiridos.

2.3.1.5 Módulo Hoja de Visita

Permite al vendedor llevar un registro de cobros efectuados en cada visita. El módulo incluye información referente a la fecha de la visita, el nombre del cliente, forma de pago, así posteriormente se podrá llevar la retroalimentación sobre la actividad del vendedor en cuanto a los cobros realizados.

2.3.1.6 Módulo Reportes

Permite consultar la información sobre los cobros efectuados, registro de nuevos clientes además de consultar el detalle de la cartera de todos los clientes activos y el reporte de todos los pedidos efectuadas desde el móvil incluyendo su ubicación.

2.3.2 Limitaciones

- La aplicación no permitirá modificar los datos del cliente, ya que la información personal se encuentra almacenada en el servidor central de la empresa y podrá ser modificada solo desde allí.
- El vendedor no podrá escoger ítems de diferentes combos ya que se maneja el concepto de líneas de productos agrupados por categorías.
- La actualización estará limitada a realizarse siempre y cuando se cuente con una conexión a internet.
- La aplicación móvil no implementará el módulo facturación, se limita a realizar el registro de una orden de pedido offline.

2.3.3 Stakeolders

ID	Nombre	Instrucción	Cargo	Categoría	Relevancia
001	Ing. Cristian Timbi	Superior	Tutor	Supervisor	5
002	Ing. Francisco Maldonado	Superior	Equipo de Supervisión "AGROTA CIA. LTDA"	Jefe del Área de Sistemas.	5
003	Johanna Picón	Superior	Investigador Programador	Analista	5
004	Sebastián Zhinin	Superior	Investigador Programador	Analista	5

Tabla 19. Especificación de Stakeolders. **Fuente:** El autor.

2.4 Selección de las herramientas de soporte y desarrollo

Un enfoque móvil presenta varias limitantes para el desarrollador parámetros como la conexión a internet, ubicación y duración de batería nos obliga a elegir los mejores recursos para simplificar los esfuerzos de programación.

El kit de desarrollo de Android contiene una amplia variedad de herramientas y componentes de plataforma que aportan simplicidad de instalación, reutilización de código, facilidad de documentación; predominado la ligereza de las librerías sobre todo lo demás.

A continuación se presenta algunos paquetes propios de Android que facilitan la creación de servicios, gestión de red, localización, etc.

- **Paquete android.app.service:** contiene los servicios de la aplicación, es decir el conjunto de tareas que se ejecutan sin la intervención del usuario. Un servicio en la aplicación móvil tiene doble funcionalidad ya sea *"permitir la ejecución de código en segundo plano o funcionar como un mecanismo de comunicación entre aplicaciones"* [1]. Un ejemplo de servicio que corre en segundo plano es la galería de imágenes utilizado para descargar imágenes .bmp directamente del servidor.

- **Paquete android.net.wifi:** contiene el API de conectividad Wifi, incluye la clase WifiManager encargada de administrar todos los recursos inalámbricos. Podemos obtener una instancia de ésta clase llamando al método *getSystemService.* Su sintaxis es la siguiente:

```
WifiManager mainWifiObj;
mainWifiObj = (WifiManager) getSystemService (Context.WIFI_SERVICE);
```

Ilustración 20. Sintaxis para invocar al método getSystemService [44].

Si deseamos escanear todas las redes que se encuentren a nuestro alrededor podemos llamar al método *StartScan*, el cual nos devolverá una lista de objetos ScanResult. Su sintaxis es la siguiente:

```
Lista <ScanResult> wifiScanList = mainWifiObj.getScanResults ();
Datos String = wifiScanList.get (0) .toString ()
```

Ilustración 21. Sintaxis para invocar al método SatrtScan [44].

- **Paquete android.locate.manager:** Aporta simplicidad para desarrollar aplicaciones de Geolocalización ya que permite obtener coordenadas de latitud, longitud a través de la red GPS o mediante la consulta a la base de datos de redes Wifi [45].

CAPÍTULO 3

3.DISEÑO DEL SOFTWARE

Introducción

La etapa de diseño es la más importante y definitoria dentro del proceso de desarrollo de software. Esta etapa consiste a grandes rasgos en aplicar diferentes técnicas y herramientas con el fin de obtener diagramas suficientemente detallados como para que cualquier persona pueda entender la funcionalidad del sistema.

La Descripción de Diseño del Software (DDS) representa el núcleo técnico de la ingeniería del software. Una vez que se ha determinado los requerimientos el siguiente paso es definir la arquitectura, paquetes, diseño de interfaces y otras características del sistema de una forma más estandarizada.

Propósito

El presente capítulo tiene como propósito mostrar la estructura de la aplicación móvil mediante la implementación de diagramas de clases, diagramas de paquetes, diagramas de casos de uso, los mismos que se detallan en la SRS.

El documento de diseño proporciona al equipo de desarrollo una orientación general de la arquitectura de la aplicación y las consideraciones técnicas que se debe tener para diseñar un producto de software robusto, escalable y portable.

El objetivo de la fase de diseño es crear un documento soporte que contenga la información sobre el diseño de datos, diseño arquitectónico, diseño de casos de uso, de tal forma que el programar tenga un marco de referencia y sugerencias sobre las buenas prácticas de diseño que se deben seguir a la hora de crear una aplicación móvil empresarial.

El siguiente documento contempla algunas recomendaciones del estándar IEEE 1016-2009, presentando una breve descripción de los paquetes del sistema, estilo arquitectónico, vistas arquitectónicas, etc. Una vez que el documento de diseño sea aprobado por los stakeolders, se convertirá en una línea base para limitar los cambios en el diseño del proyecto.

3.1 Diseño de la arquitectura solución

El proceso de diseño de la arquitectura implica un análisis riguroso del objetivo que persigue el negocio, la decisión más acertada será la que solucione tanto los requisitos técnicos como operacionales planteados por el cliente.

Considerando que la empresa cuenta con una infraestructura actual, maneja un gran número de clientes y desea promover movilidad a sus vendedores. La solución planteada para Agrota Cía. Ltda, se basa en una arquitectura tres capas. Cada capa representa un proceso separado y bien definido que facilita el intercambio de información entre los diferentes niveles.

- **Nivel 1 -Capa de Presentación:** Representa una fachada ya que su función principal es crear un entorno amigable para el usuario, además de encargarse del manejo de ventanas, menús, gráficos, etc. Es considerada la parte más trabajosa para el desarrollador ya que el hecho que una aplicación cuente con una interfaz de usuario pobre nunca permitirá satisfacer las expectativas del negocio.

  La interfaz de usuario constituye la administración visual de los datos, obtenidos mediante el uso de objetos de la capa de negocio [47]. Un error frecuente es tratar de crear reglas de negocio en el interfaz de usuario, esto sin lugar a duda impediría la reutilización y despliegue de la aplicación.

- **Nivel 2- Manejo del negocio:** Cuando las aplicaciones adquieren cierta complejidad y la lógica de acceso a datos no es suficiente para encapsular toda la información. La forma mas sencilla de solucionar el inconveniente es crear una capa intermedia.

  Su función será aislar la capa de presentación de la capa del servidor, de forma que las estructuras de datos y la lógica sean independientes y que la detección errores y mantenimiento sea más sencillo [47].

La capa de negocio soportará todas las normas de acceso a datos,es por esta razón que constituye el grueso de la lógica del funcionamiento de la aplicación móvil.

- **Nivel 3-Servidor:** Incluye los objetos de acceso a datos que son los intermediarios entre la aplicación y los orígenes de datos[48]. Estos objetos son los únicos encargados de conectarse con la base de datos central, enviarles sentencias SQL y órdenes de ejecución.

  La función principal de esta capa es recuperar y almacenar los datos con los que trabaja la aplicación mediante un servicio web. Estos servicios son procesos que se mantienen ejecutándose en equipos dedicados a la escucha de alguna solicitud.

La Ilustración 22 presenta la arquitectura de la aplicación WISE dentro del marco de estudio diseño de una aplicación móvil en n capas:

- ✓ **Capa de Presentación:** El vendedor interactuará con la capa de presentación al acceder al menú principal de la aplicación para registrar una orden de pedido offline. Una vez que el pedido sea validado será almacenado en la base de datos móvil para su posterior procesamiento.

- ✓ **Capa Lógica de Negocio:** Recibe la petición del vendedor y se encarga de aplicar la lógica de tratamiento de datos y las normas de acceso para facilitar el intercambio de información con el servidor central.

- ✓ **Capa Acceso a la Base de datos:** Compuesta por la base de datos se encarga del almacenaciento, concurrencia y validacion de información. Incluye también los servicios web responsables de la comunicación entre diferentes aplicaciones.

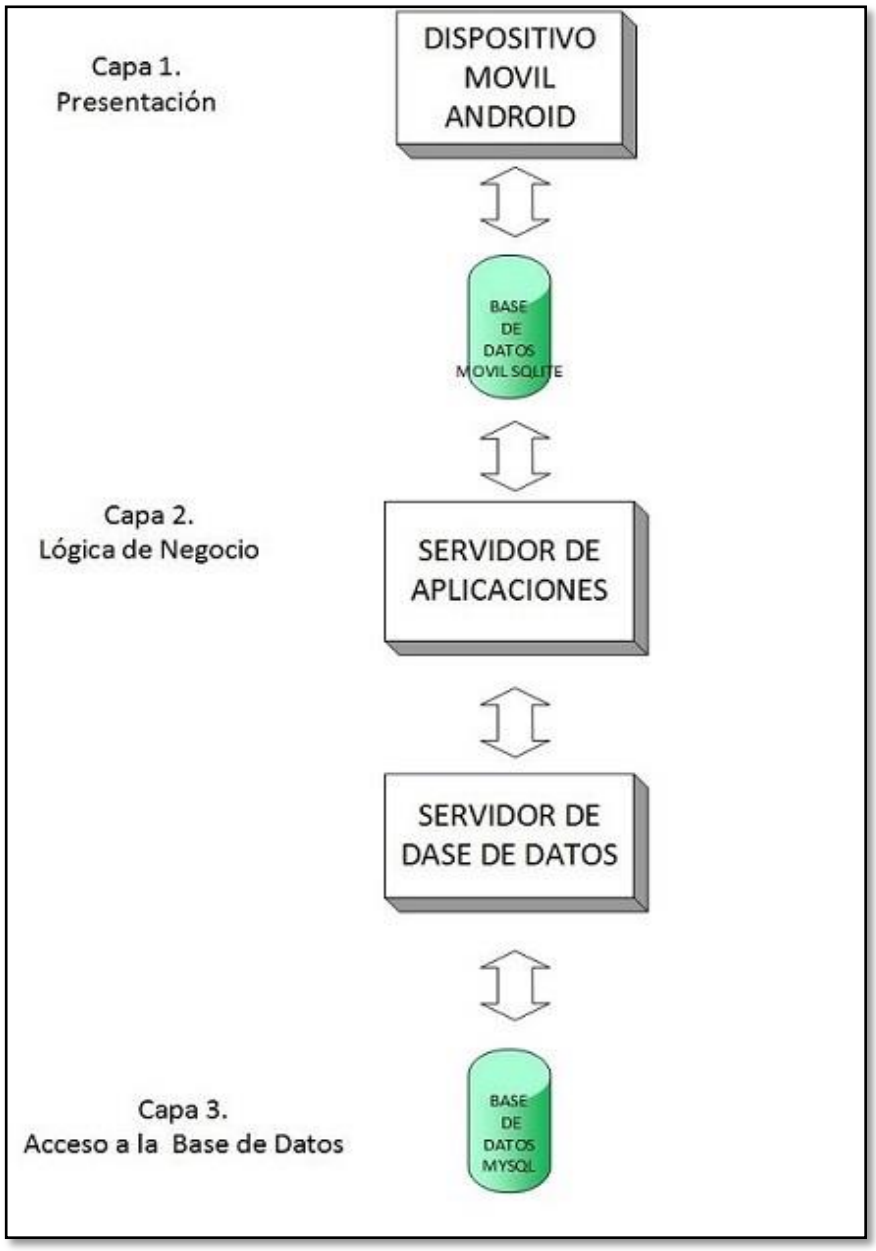


Ilustración 22. Arquitectura de la aplicación móvil WISE. **Fuente:** El autor.

3.1.1 Descripción General de la Arquitectura del Sistema

La arquitectura tres capas permite integrar dos sistemas que funcionan en diferentes plataformas. Por un lado tenemos el sistema central basado en Visual Studio .net y por otro la aplicación móvil desarrollada en Android. A continuación se detallará la funcionalidad de cada capa que conforma la aplicación móvil.

Ilustración 23. Descripción de la Arquitectura del sistema. **Fuente:** El autor.

3.2 Documentación UML

"UML es un lenguaje que permite modelar, construir y documentar los elementos que forman un sistema de software orientado a objetos" [49]. Se ha convertido en un estándar de facto de la industria debido a la incorporación de buenas prácticas de diseño.

"La estandarización de un lenguaje de modelado es invaluable, ya que es la parte principal del proceso de comunicación que requieren todos los involucrados en un proyecto" [50]. Si se quiere discutir modificaciones en un diseño,ambas partes deben conocer el lenguaje de modelado base.

Desde el punto de vista del desarrollador UML representa una hoja de ruta que permite medir el progreso del proyecto y aportar visibilidad al cliente.

3.2.1 Descripción de los casos de uso

Un diagrama de casos de uso muestra, los distintos requisitos funcionales que se esperan de una aplicación y como se relacionan con su entorno (usuarios u otras aplicaciones). Todo sistema tiene como mínimo un diagrama de caso uso que representa el entorno del sistema (actores) y su funcionalidad principal (casos de usos).

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

- **Actor:** Representa un rol que es llevado a cabo por una persona. Un actor en un diagrama de caso es representado por una figura en forma de persona.

- **Caso de Uso:** Es una tarea que debe poder llevarse a cabo con el apoyo del sistema que se está desarrollando, se representa mediante un ovalo.

- **Comunicación:** Representa la relación existente entre un caso de uso y un actor, dicho elemento es representado por una línea recta que se extiende de la figura del actor hacia el ovalo.

3.2.2 Requisitos para los casos de uso

La aplicación móvil ha definido los siguientes actores :

- **Administrador:** Posee todos los privilegios para crear,modificar o borrar perfiles de usuarios.

- **Vendedor:** Se desplaza por la ciudad llevando consigo un dispositivo móvil en el que se encuentra instalado la aplicación,permitiendo realizar el registro de pedidos offline, consulta de la cartera del cliente,consulta de catálogos de productos, emitir el registo de vista,etc.

3.2.3 Diagramas de Caso de Uso

Los casos de uso ayudan al programador a extraer y organizar los requerimientos impuestos por el cliente además de crear un punto de partida para iniciar la labor de desarrollo. A continuación se muestra el modelo de caso de uso global del sistema móvil presentando una visión detallada de la interacción de los actores con el sistema. Para mayor información consulte el Anexo A, el cual ha sido diseñada bajo el estándar IEEE [2] 1016-2009 [64]. Indicando la relación entre los requerimientos funcionales incluidos en el capítulo 2 y los casos de uso diseñados a continuación:

[2] IEEE (Institute Of Electrical And Electronical Engineers)

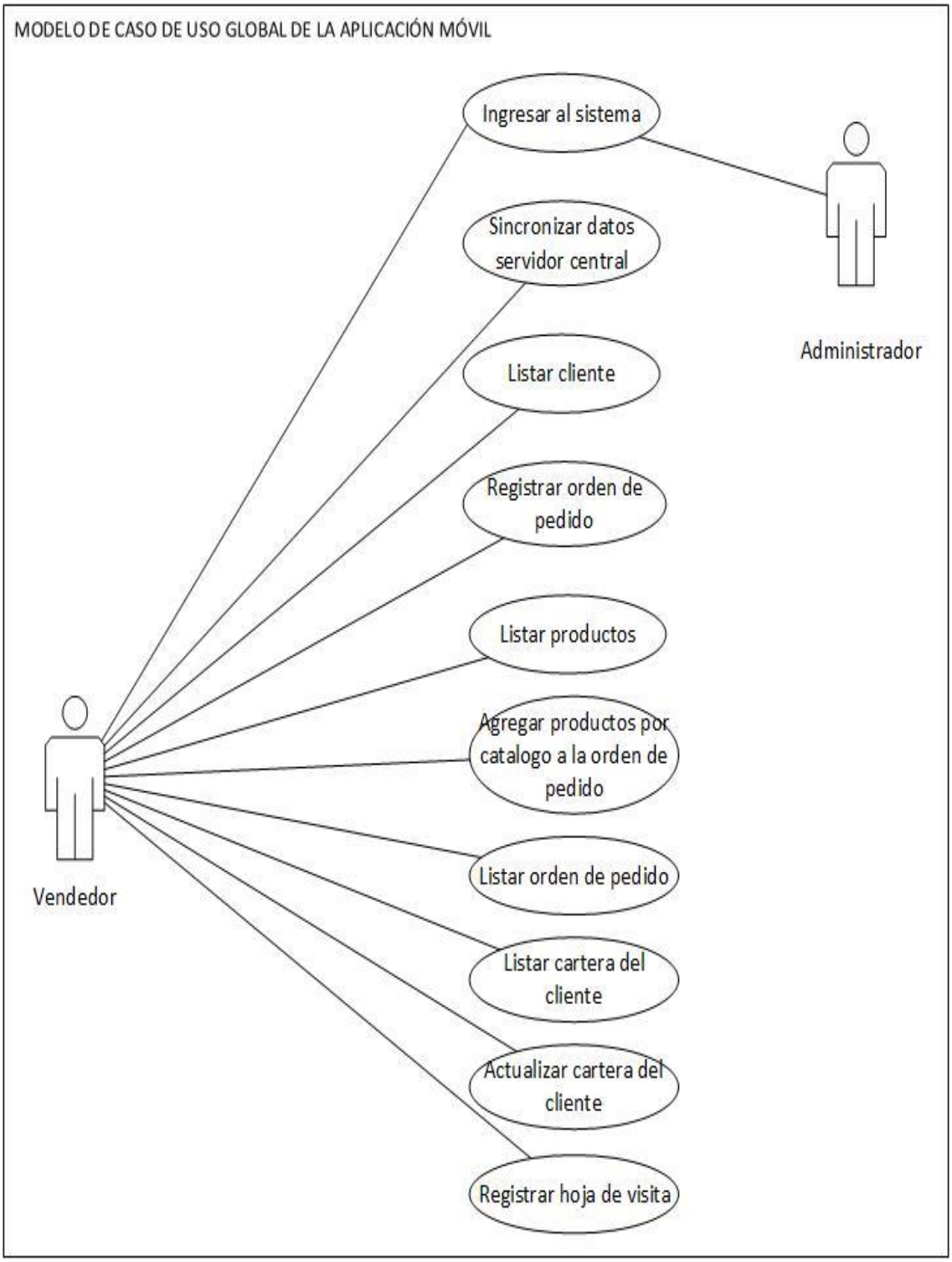
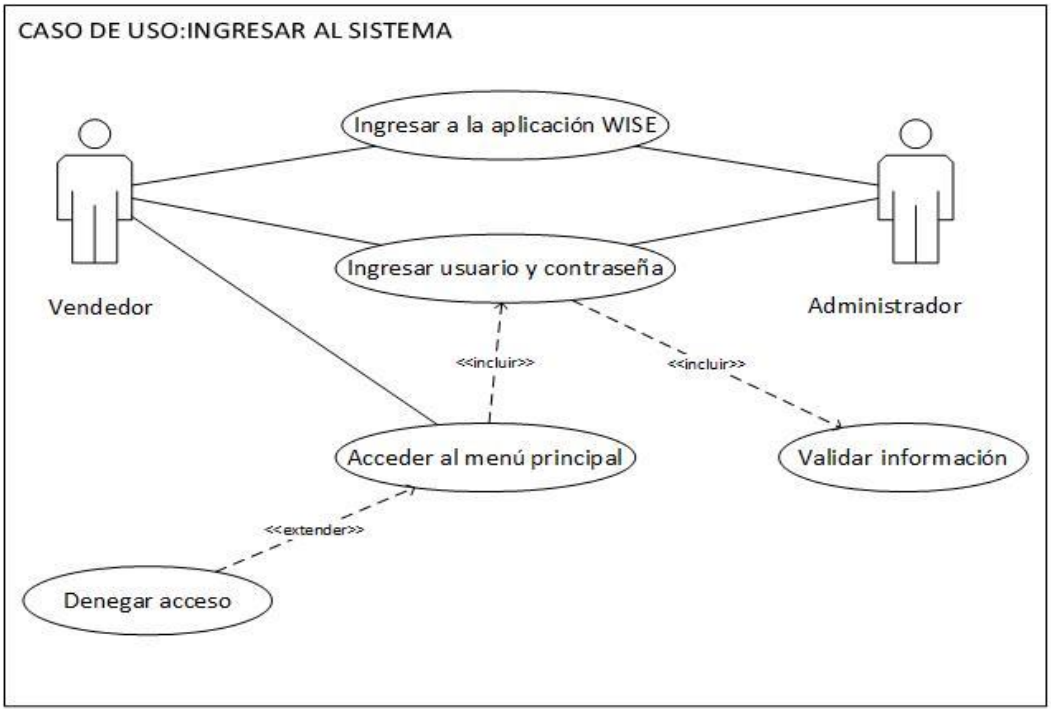


Ilustración 24. Diseño de Caso de Uso Global de la aplicación móvil. **Fuente:** El autor.

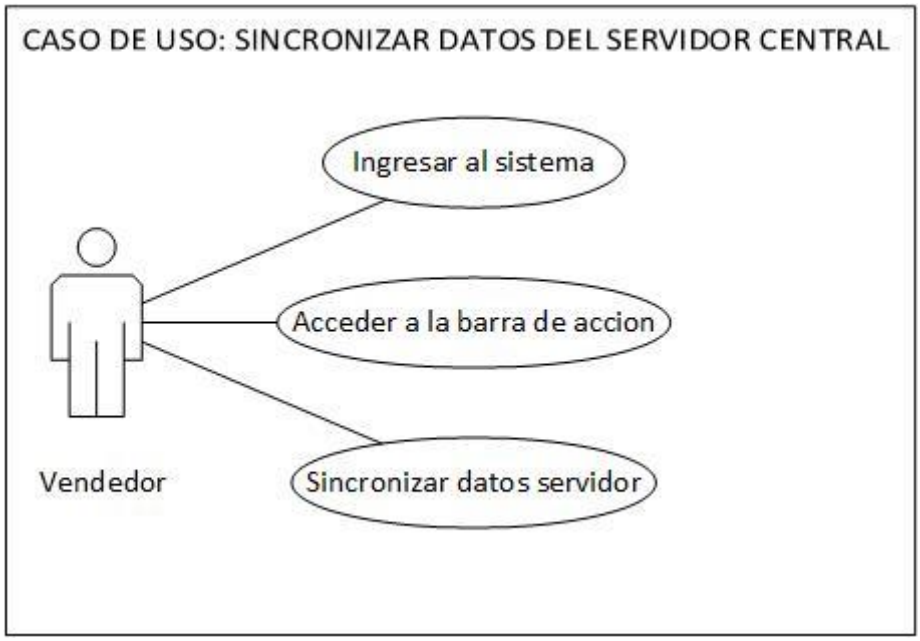
El siguiente paso consiste en descomponer cada caso en partes más pequeñas que permitan entender el flujo de eventos y la forma en la que los usuarios interactúan con la aplicación.



MODELO DE DESCRIPCIÓN	
CU- 001:	Ingresar al sistema
Autores:	Johanna Picón – Sebastián Zhinin
RELACIONES	
Descripción:	La aplicación WISE solo permite el ingreso mediante un usuario y contraseña previamente creado por el administrador en el servidor central.
Pre-condición:	El usuario ha ingresado a la aplicación mediante el sistema de autenticación.
Pos-condición:	La aplicación WISE carga la pantalla del Menú Principal la misma que incluye los módulos: Módulo Pedidos, Módulo Cliente, Módulo Cartera, Módulo Producto, Módulo Hoja de Visita y Módulo Reportes.
Actor Primario:	Vendedor
Actor Secundario:	Administrador

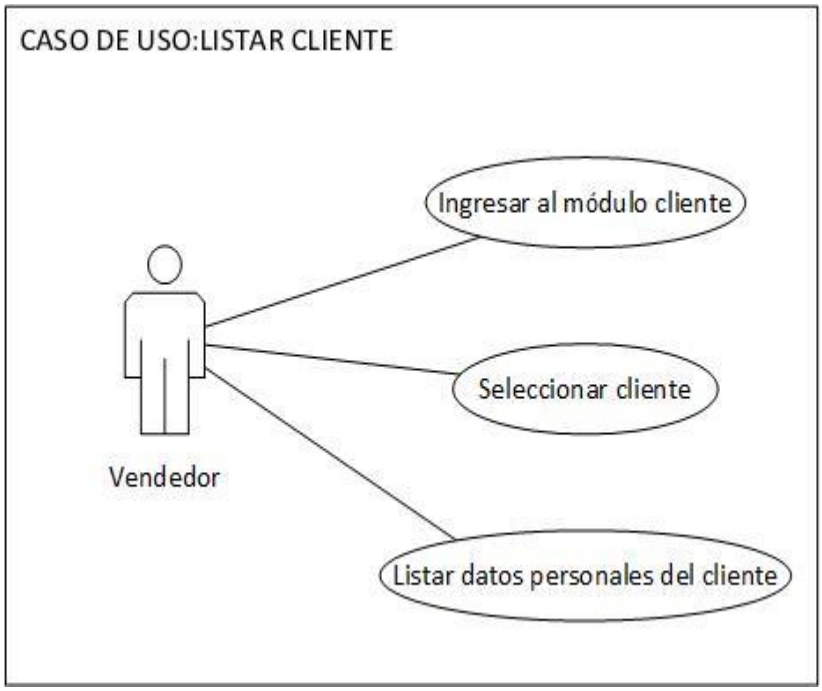
FLUJO DE EVENTOS		
Intenciones del usuario	**Responsabilidad de la Aplicación**	**Excepciones**
1. El usuario ingresa su credencial (login y contraseña) para poder acceder a la aplicación WISE y utilizar los servicios ofrecidos.	**2.** La aplicación móvil valida la información realizando una llamada al servidor central mediante los web Services. **3.** Si la autentificación es correcta se permite el acceso caso contrario será denegado cuando:	**4.** El vendedor ingrese información incorrecta. **5.** El usuario no se encuentra registrado en la base de datos del servidor central. **6.** El dispositivo móvil no ha sido registrado previamente.

Tabla 20. Descripción Caso de Uso: Ingresar al Sistema. **Fuente:** El autor.



MODELO DE DESCRIPCIÓN		
CU- 002:	Sincronizar datos del servidor central	
Autores:	Johanna Picón – Sebastián Zhinin	
RELACIONES		
Descripción:	La aplicación WISE permite sincronizar todos los datos nuevos o modificados en el servidor.	
Pre-condición:	Ingresar al sistema	
Pos-condición:	Acceder a la opción sincronizar ubicada en la barra de acción.	
Actor Primario:	Vendedor	
FLUJO DE EVENTOS		
Intenciones del usuario	**Responsabilidad de la Aplicación**	**Excepciones**
1. Verificar conexión a internet.	**2.** Siempre y cuando la aplicación cuente con acceso a internet, se podrá realizar la primera sincronización con el servidor.	**3.** El dispositivo móvil no cuenta con una conexión a internet para poder iniciar el proceso de sincronización.

Tabla 21. Descripción Caso de Uso: Actualizar datos del servidor central. **Fuente:** El autor.



MODELO DE DESCRIPCIÓN		
CU- 003:	Listar Cliente	
Autores:	Johanna Picón – Sebastián Zhinin	
RELACIONES		
Descripción:	La aplicación móvil permite listar la información del cliente.	
Pre-condición:	Ingresar al menú principal de la aplicación WISE.	
Pos-condición:	Ingresar al módulo cliente.	
Actor Primario:	Vendedor	
FLUJO DE EVENTOS		
Intenciones del usuario	**Responsabilidad de la Aplicación**	**Excepciones**
1. Buscar cliente **2.** Consultar la información personal del cliente.	**3.** Permitir al vendedor visualizar la información del cliente. **4.** La información del cliente contendrá la siguiente información de detalle: nombres,apellidos,nombreComercial, teléfono, email, cartera, calificación y ubicación.	**5.** El cliente no se encuentra registrado en el servidor central.

Tabla 22. Descripción Caso de Uso: Listar Cliente. **Fuente:** El autor.

MODELO DE DESCRIPCIÓN		
CU- 004:	Registrar Orden de Pedido	
Autores:	Johanna Picón – Sebastián Zhinin	
RELACIONES		
Descripción:	Registrar una orden de pedido independientemente de contar con una conexión a internet.	
Pre-condición:	Ingresar al sistema	
Pos-condición:	Escoger un cliente y acceder al módulo pedido.	
Actor Primario:	Vendedor	
FLUJO DE EVENTOS		
Intenciones del usuario	**Responsabilidad de la Aplicación**	**Excepciones**
1. Seleccionar un cliente **2.** Ingresar al módulo orden de pedido. **3.** Buscar ítems dentro de un catálogo. **4.** Seleccionar el item padre. **7.** Editar la cantidad a comprar. **8.** Aplicar forma de pago al pedido. **9.** Ingresar el costo de transporte. **10.** Ingresar las observaciones necesarias.	**5**. Si el ítem tiene extras se visualizará el combo asociado. **6.** Si el item incluye una promoción Se visualizará todas las promociones vigentes. **11.**Validar información **12**. Grabar el registro de pedido y enviarlo al servidor central.	**13.** La aplicación solo permite escoger un ítem de cada grupo del combo, caso contrario se emitirá un mensaje de error. **14.** Si la orden de pedido no se ha guardado satisfactoriamente se emitirá un mensaje indicando el inconveniente.

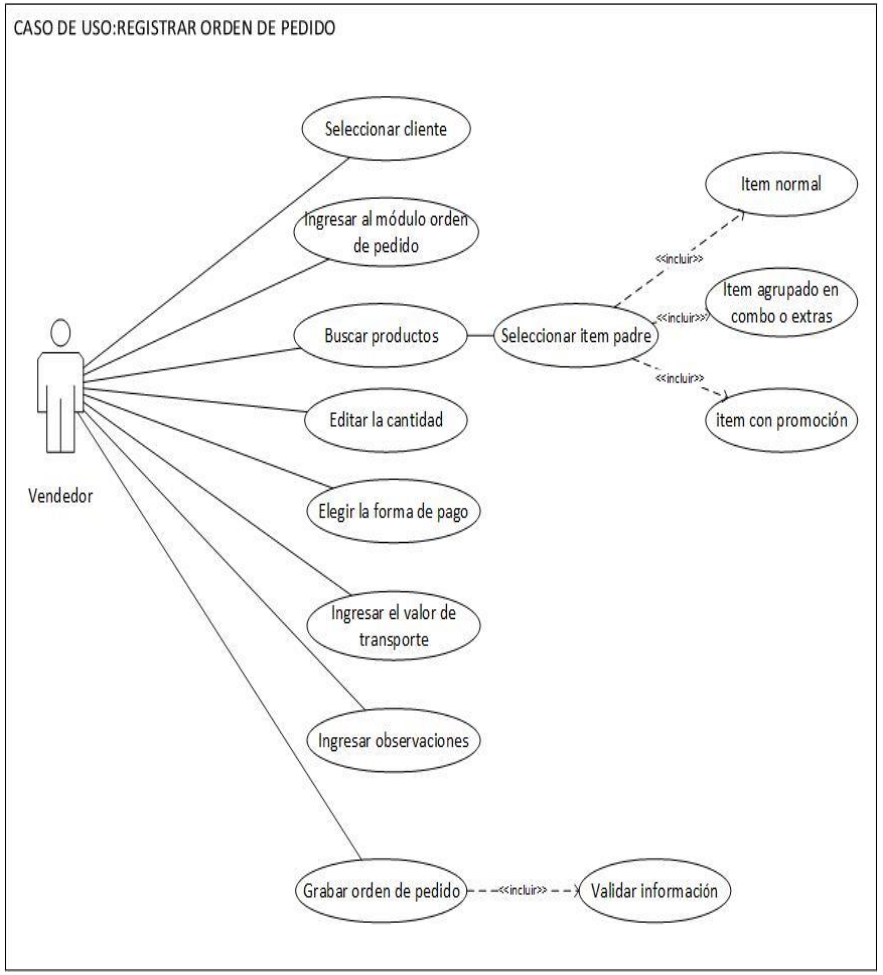
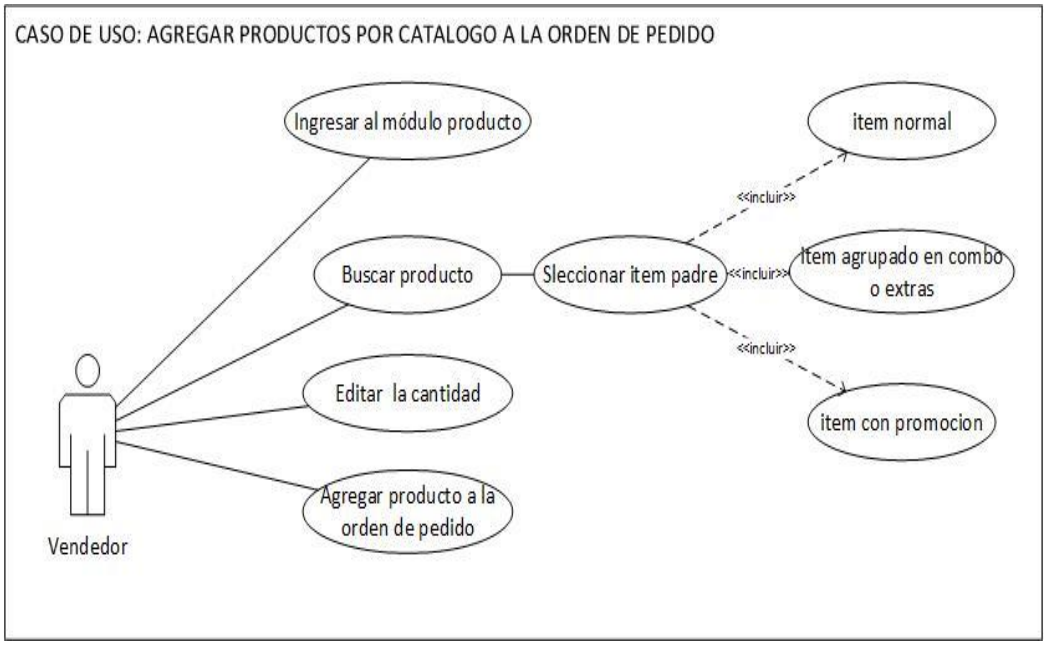


Tabla 22. Descripción Caso de Uso: Registrar Orden de Pedido. **Fuente:** El autor.

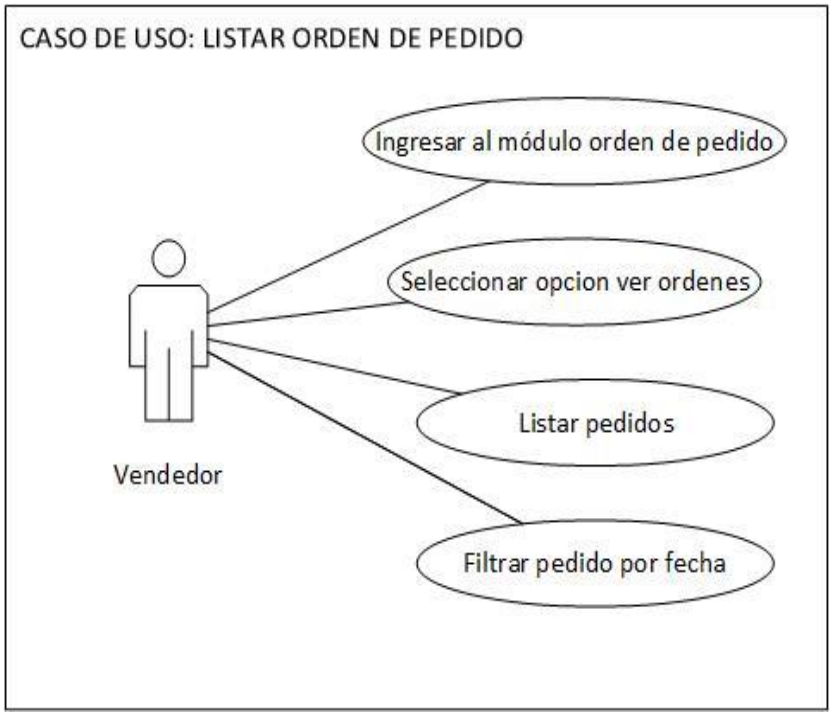
MODELO DE DESCRIPCIÓN		
CU- 005:	Listar Productos	
Autores:	Johanna Picón – Sebastián Zhinin	
RELACIONES		
Descripción:	La aplicación móvil permite listar productos pertenecientes a una categorización.	
Pre-condición:	El vendedor ingresa al módulo producto y busca los productos de acuerdo a una línea, categoría o marca.	
Pos-condición:	Se visualizará una lista con todos productos pertenecientes a esa categorización.	
Actor Primario:	Vendedor	
FLUJO DE EVENTOS		
Intenciones del usuario	**Responsabilidad de la Aplicación**	**Excepciones**
1. Buscar productos por una categorización.	**2.** Permitir filtrar productos por líneas, categorías o marcas. **3.** Emitir una lista de productos con la siguiente información de detalle: nombreProducto, descuento, cliente, imagen.	**4.** No existen productos disponibles en esa categorización.

Tabla 23. Descripción Caso de Uso: Listar Productos. **Fuente:** El autor.



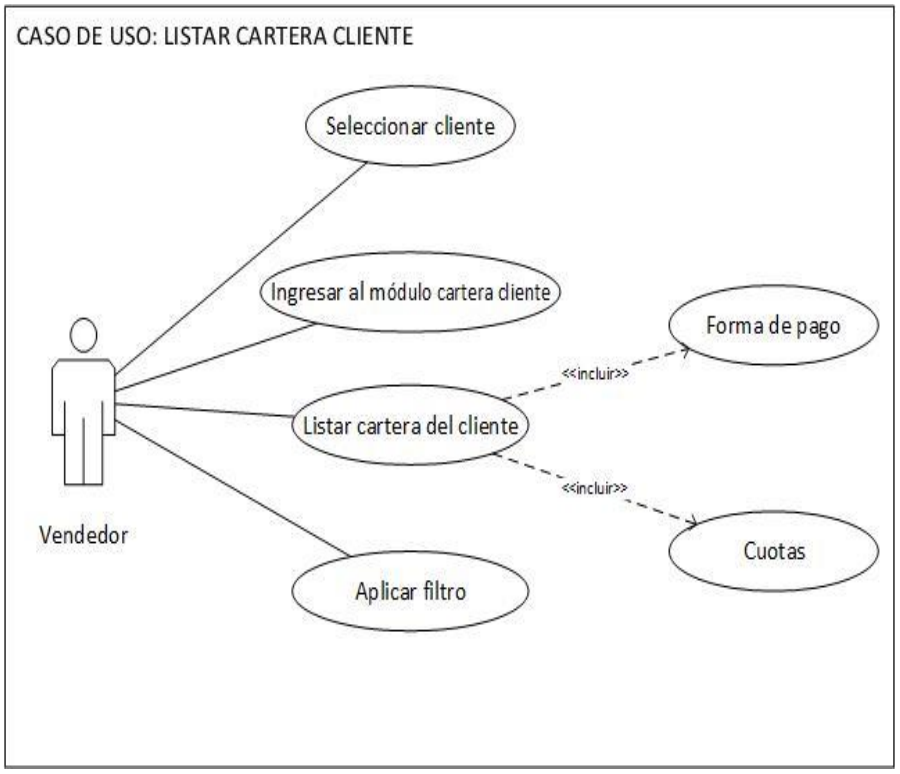
MODELO DE DESCRIPCIÓN		
CU- 006:	Agregar productos por catálogo a la orden de pedido.	
Autores:	Johanna Picón – Sebastián Zhinin	
RELACIONES		
Descripción:	La aplicación WISE permite agregar productos por catálogo directamente a la orden de pedido.	
Pre-condición:	El vendedor ingresa al módulo producto y pulsa el botón agregar producto.	
Pos-condición:	El vendedor edita la cantidad deseada.	
Actor Primario:	Vendedor	
FLUJO DE EVENTOS		
Intenciones del usuario	**Responsabilidad de la Aplicación**	**Excepciones**
1. Buscar productos de acuerdo a una categorización específica. **2.** Seleccionar el item padre. **5.** Editar la cantidad deseada.	**3.** Si el ítem tiene extras se visualizará el combo asociado. **4.** Si el item incluye una promoción se visualizará todas las promociones vigentes. **6.** Agregar productos a la orden de pedido.	**7.** Se debe agregar por lo menos un item a la orden de pedido.

Tabla 24.Descripción Caso de Uso: Agregar productos por catálogo a la orden de pedido.
Fuente: El autor.



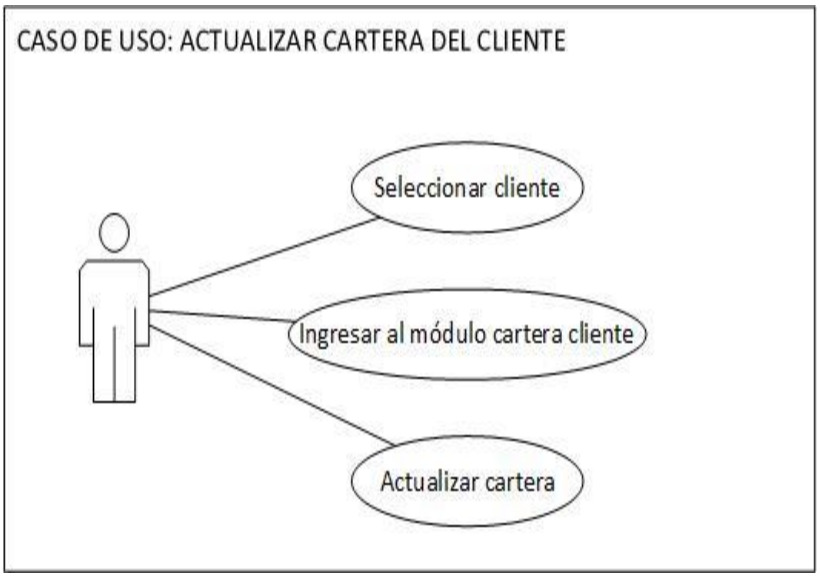
MODELO DE DESCRIPCIÓN		
CU- 007:	Listar Orden de Pedido	
Autores:	Johanna Picón – Sebastián Zhinin	
RELACIONES		
Descripción:	La aplicación permite listar todos los pedidos registrados en la base de datos móvil de acuerdo a una fecha o cliente especifico.	
Pre-condición:	Ingresar al menú principal de la aplicación WISE.	
Pos-condición:	Ingresar al módulo pedido.	
Actor Primario:	Vendedor	
FLUJO DE EVENTOS		
Intenciones del usuario	**Responsabilidad de la Aplicación**	**Excepciones**
1. Seleccionar la opción ver pedido. **2.** Ingresar la fecha o cliente para efectuar la búsqueda del pedido.	**3.** Permitir visualizar una lista de pedidos emitidos en una fecha o por cliente. **4.** Cada pedido contendrá la siguiente información de detalle: numeroOrden, clientes, fecha, valor, estado.	**5.** Si no existe almenos un pedido creado en esa fecha, se emitirá un mensaje indicándolo.

Tabla 25. Descripción Caso de Uso: Listar Orden de Pedido. **Fuente:** El autor.



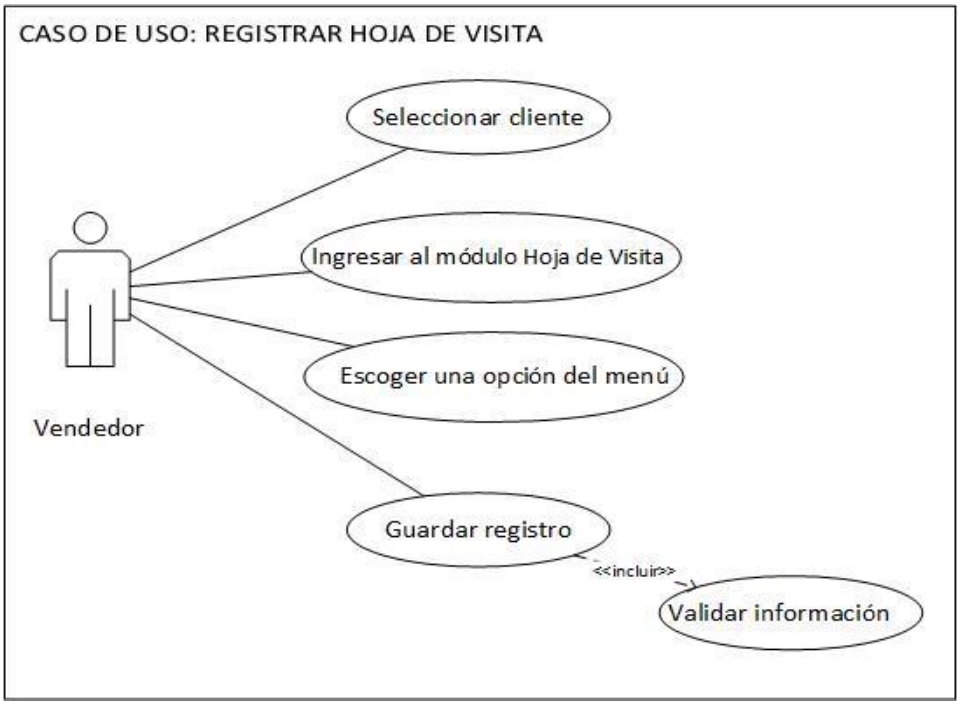
MODELO DE DESCRIPCIÓN		
CU- 008:	Listar Cartera del Cliente.	
Autores:	Johanna Picón – Sebastián Zhinin	
RELACIONES		
Descripción:	La aplicación móvil permite listar toda la información que contiene la cartera del cliente.	
Pre-condición:	Ingresar al menú principal de la aplicación WISE.	
Pos-condición:	Ingresar al módulo cartera del cliente.	
Actor Primario:	Vendedor	
FLUJO DE EVENTOS		
Intenciones del usuario	**Responsabilidad de la Aplicación**	**Excepciones**
1. Buscar un cliente y consultar el estado de su cartera.	**2.** Emitir una lista indicando toda la información que incluye la cartera del cliente. **3.** La cartera del cliente contendrá el siguiente detalle: facturas, fecha, valor, retención, saldo, forma de pago y cuotas.	**4.** El cliente no tiene una cartera activa.

Tabla 26. Descripción Caso de Uso: Listar cartera del cliente. **Fuente:** El autor.



MODELO DE DESCRIPCIÓN		
CU- 009:	Actualizar Cartera del Cliente	
Autores:	Johanna Picón – Sebastián Zhinin	
RELACIONES		
Descripción:	La aplicación WISE permite actualizar la información que contiene la cartera del cliente.	
Pre-condición:	Ingresar al sistema	
Pos-condición:	Seleccionar un cliente	
Actor Primario:	Vendedor	
FLUJO DE EVENTOS		
Intenciones del usuario	**Responsabilidad de la Aplicación**	**Excepciones**
1. Ingresar al módulo cartera cliente **2.** Pulsar el botón actualizar cartera.	**3.** Actualizar la información de la cartera del cliente, esto incluye: facturas emitidas y el detalle del crédito (forma de pago y cuotas).	**4.** El dispositivo móvil no cuenta con una conexión a internet.

Tabla 27. Descripción Caso de Uso: Actualizar cartera del cliente. **Fuente:** El autor.



MODELO DE DESCRIPCIÓN		
CU- 009:	Registrar hoja de visita	
Autores:	Johanna Picón – Sebastián Zhinin	
RELACIONES		
Descripción:	La aplicación WISE permite registrar los cobros, clientes nuevos y el estado de la visita realizada por el vendedor.	
Pre-condición:	Ingresar al sistema	
Pos-condición:	Seleccionar un cliente	
Actor Primario:	Vendedor	
FLUJO DE EVENTOS		
Intenciones del usuario	**Responsabilidad de la Aplicación**	**Excepciones**
5. Ingresar al módulo hoja de visita. **6.** Seleccionar una opción del menú (registro de visita, cobros o clientes nuevos.)	**7.** Guardar la información de registro de visita, cobros o clientes nuevos ingresados en el móvil.	**8.** El vendedor no ingresa un campo obligatorio, por lo que la aplicación emite un mensaje indicando el inconveniente.

Tabla 28. Descripción Caso de Uso: Registrar Hoja de Visita. **Fuente:** El autor.

3.2.4 Vistas Arquitectónicas

El diseño del software implica un sin número de posibilidades, es prácticamente imposible obtener requerimientos de todos ellos. Es por esta razón que Philippe Kruchten plantea dos conceptos importantes: vista y punto de vista , pues bien una vista no es más que una representación gráfica de todo el sistema y un punto de vista se define como un conjunto de reglas o normas para entender la vista [46].

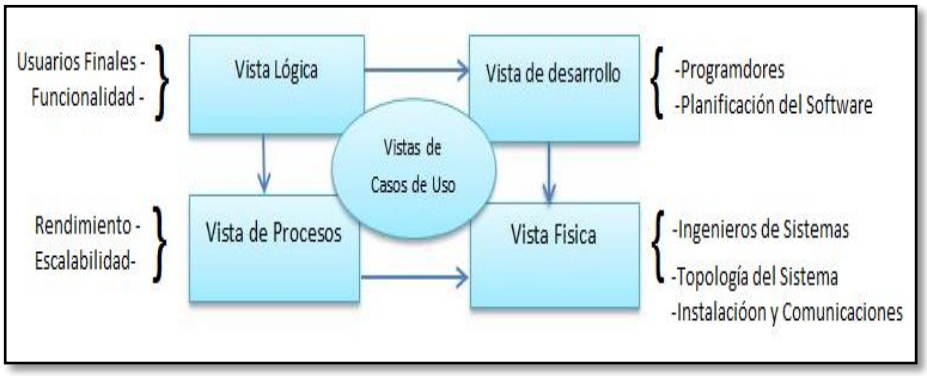


Ilustración 25. Modelo de "4+1" vistas [46].

En la Ilustración 25 podemos observar la propuesta de Kruchte de utilizar 4 diferentes vistas y 1 vista de casos de uso para validar el comportamiento del sistema. A continuación una breve explicación de cada vista:

- **La vista de procesos:** Modela la funcionalidad del sistema describiendo los mecanismos de concurrencia y sincronización. Esta vista es de gran utilidad para evaluar el rendimiento y escalabilidad del sistema.

- **La vista de desarrollo:** Permite modelar el diseño del código dentro del punto de vista del desarrollar. Tiene como finalidad contribuir a la planeación y evaluación del avance del proyecto.

- **La vista física:** Describe el mapeo del software en el hardware y refleja los aspectos de la distribución. Es muy útil para identificar la topología, requisitos de comunicación e instalación [46].

- **Los escenarios son el término +1 dentro del 4+1:** Permite abstraer los requerimientos. Su diseño se representa mediante la creación de diagramas de objetos, diagramas de casos de uso, etc. [46].

3.2.5 Punto de vista de dependencia

Permite visualizar la organización del sistema incluyendo cada uno de los componentes, tablas y paquetes que lo conforman. Uno de los principales beneficios de la vista es que permite identificar componentes que pueden compartirse con otros sistemas o entre diferentes partes de un sistema.

3.2.5.1 Vista de dependencia

La vista de dependencia constituye los paquetes principales de los que hace uso la aplicación móvil y por los cuales funcionará. La aplicación WISE ha agrupado algunas clases, métodos y atributos en seis tipos de paquetes, cada paquete representa una funcionalidad específica y una dependencia clara con cierto paquete. A continuación podemos identificar el esquema general de la aplicación WISE.

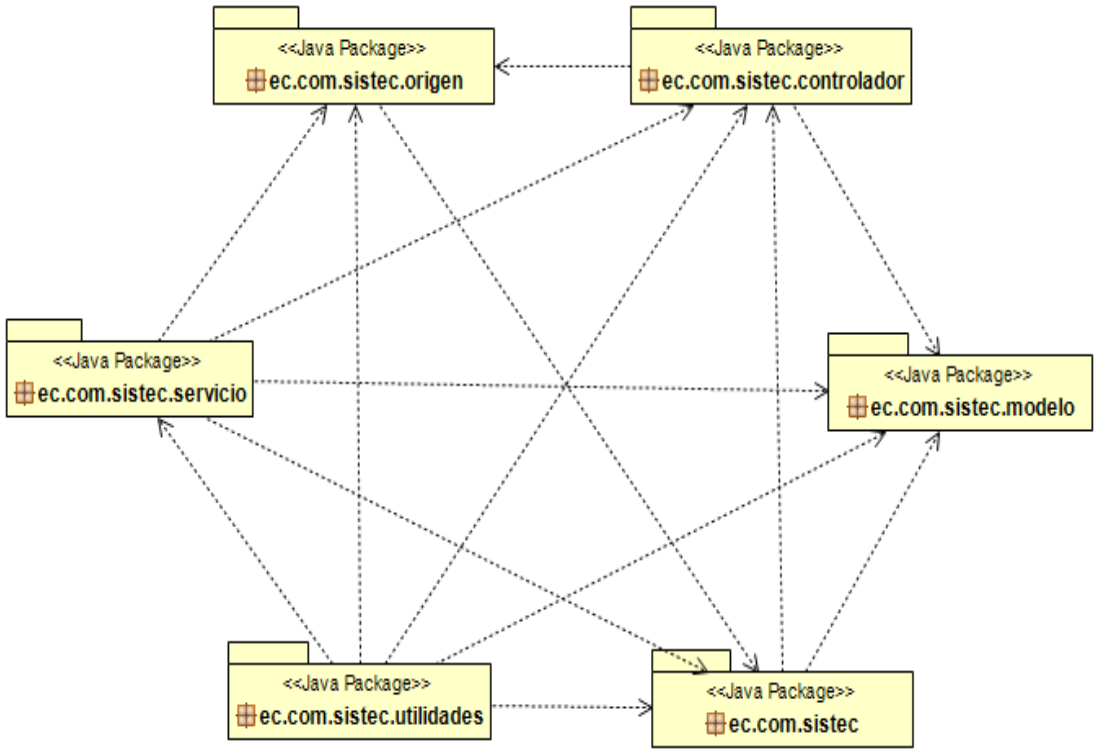


Ilustración 26. Diagrama de paquetes base de la aplicación WISE. **Fuente:** El autor.

3.2.5.2 Justificación del diseño de dependencia

El diagrama de paquetes permite identificar los diferentes módulos que conforman la aplicación y la dependencia existente entre estos. También nos presenta una forma más concisa de representación y organización del software.

Los paquetes son muy prácticos a la hora de organizar y documentar módulos grandes que pueden ser reutilizables desde otras capas. Si bien la aplicación WISE está compuesta por una variedad de clases, agrupar las diferentes funcionalidades en paquetes ayuda al desarrollador a estructurar la lógica que deberá implementar.

3.2.6 Punto de vista lógico

Un punto clave del diseño orientado a puntos de vista es buscar varias representaciones y proporcionar un marco de trabajo para descubrir conflictos en los requerimientos propuestos por los stakeolders [41]. A partir del punto de vista lógico podremos modelar las clases que componen la aplicación móvil y crear un esquema de referencia para el desarrollador.

3.2.6.1 Vista Lógica

Los diagramas de clases permiten visualizar las necesidades del usuario en un entorno de trabajo, representando la relación, notaciones y reglas entre las diferentes entidades del sistema.

Representa una expresión conceptual que permite resolver tanto los requerimientos funcionales como no funcionales, plasmando su estructura en un diseño gráfico que expresa la funcionalidad global de la aplicación móvil.

En Android todas las interfaces representan una **Activity**, estas actividades están formadas por dos partes: la parte lógica que es un archivo .java que contiene el código fuente para manipular dicha actividad y la parte gráfica que es un XML que contiene los elementos que vemos en la pantalla que son declarados mediante etiquetas.

Analizando el paquete *"ec.com.sistec"* podemos identificar varias clases Activity, las mismas que representan las ventanas con las que el vendedor deberá interactuar, estas son:

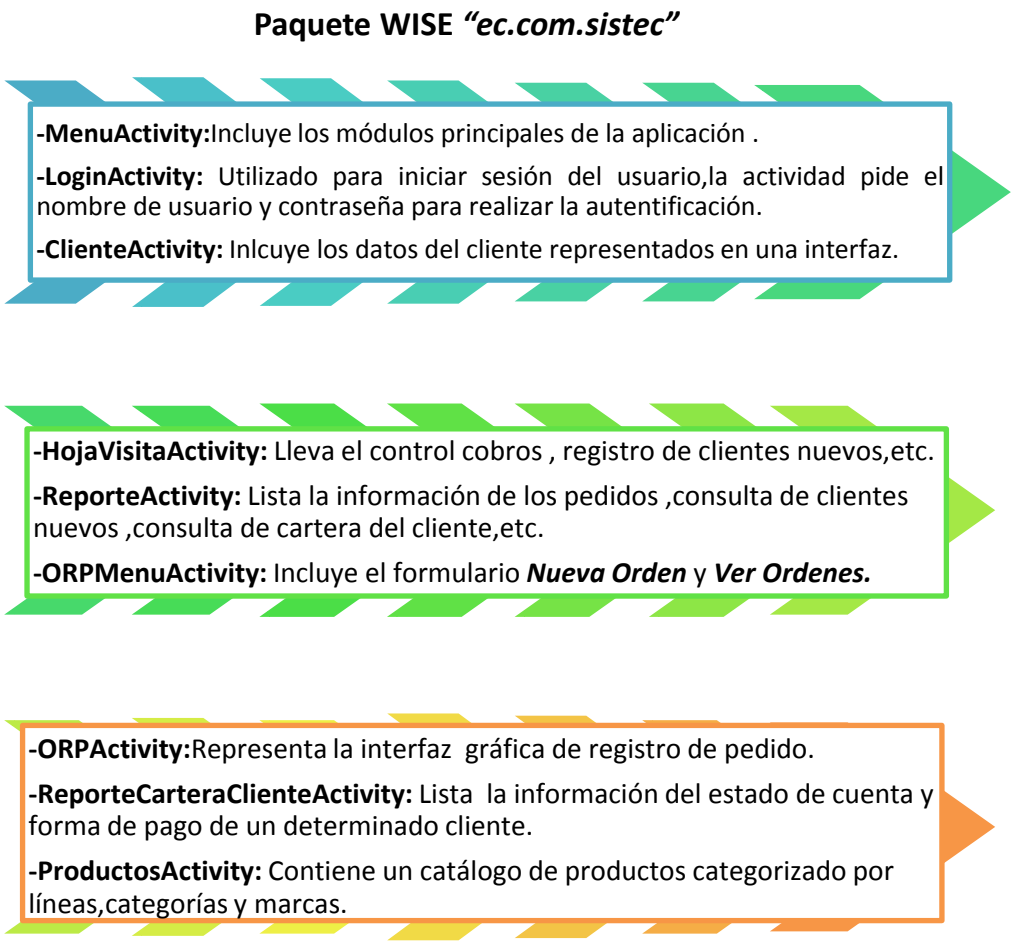


Ilustración 27. Activity que conforman el paquete ec.com.sistec. **Fuente:** El autor.

Los siguientes diagramas han recolectado la información sobre los requerimientos impuestos por las partes interesadas, aquí se ha detallado la estructura lógica de la aplicación e identificando los atributos, métodos, dependencia y relación existente entre cada una de las clases que conforman la aplicación final.

Como podemos observar en la Ilustración 28 la aplicación WISE está formada por una serie de actividades o clases Android que trabajan por un objetivo común, facilitar la navegación del usuario a través de la aplicación.

El paquete *"ec.com.sistec"* contiene las siguientes clases que conforman la interfaz gráfica de la aplicación WISE. La explicación del siguiente diagrama lo podemos encontrar en la Ilustración 27.

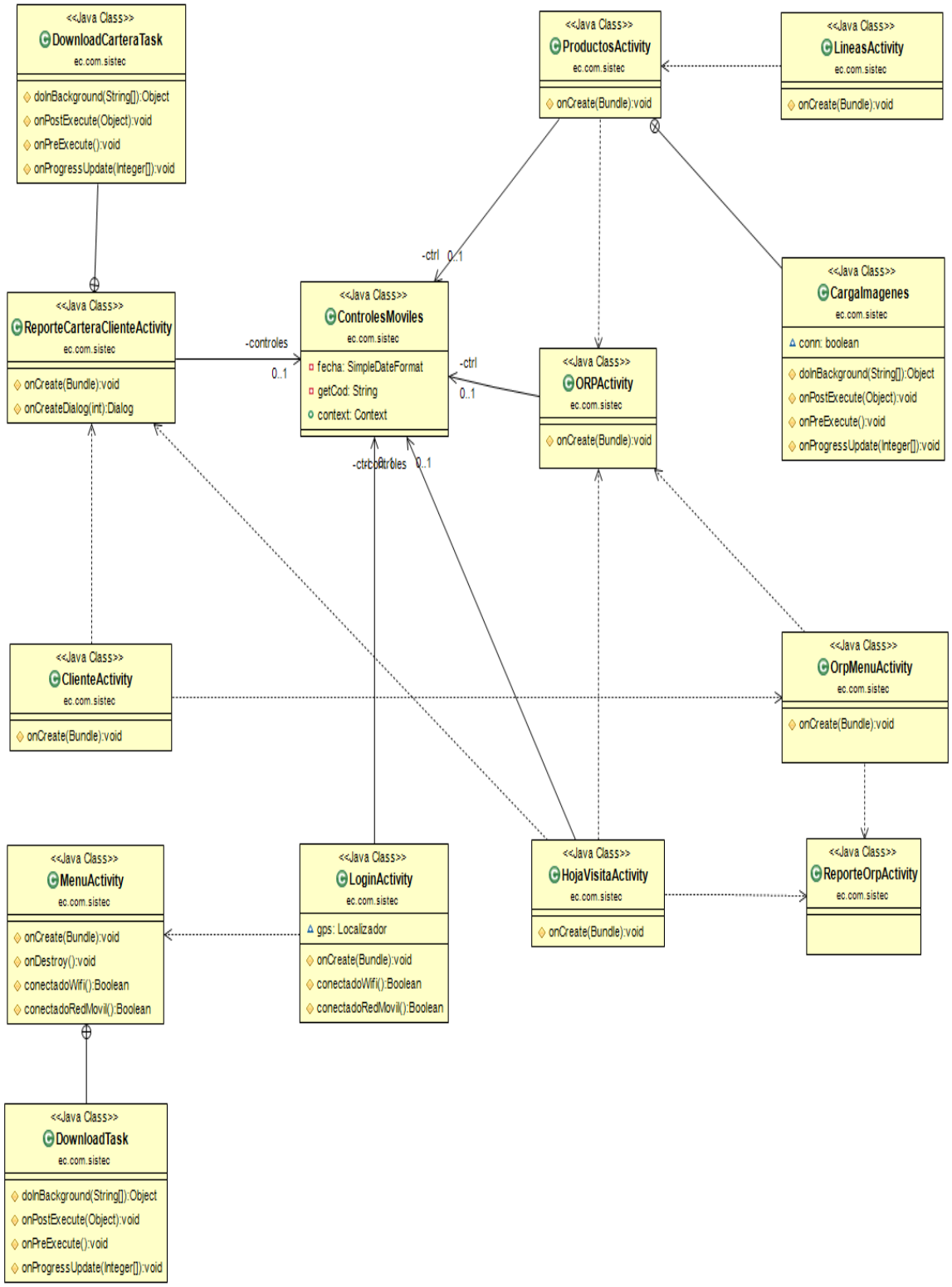


Ilustración 28 Diagrama de Clases del paquete "ec.com.sistec"**. Fuente:** El autor.

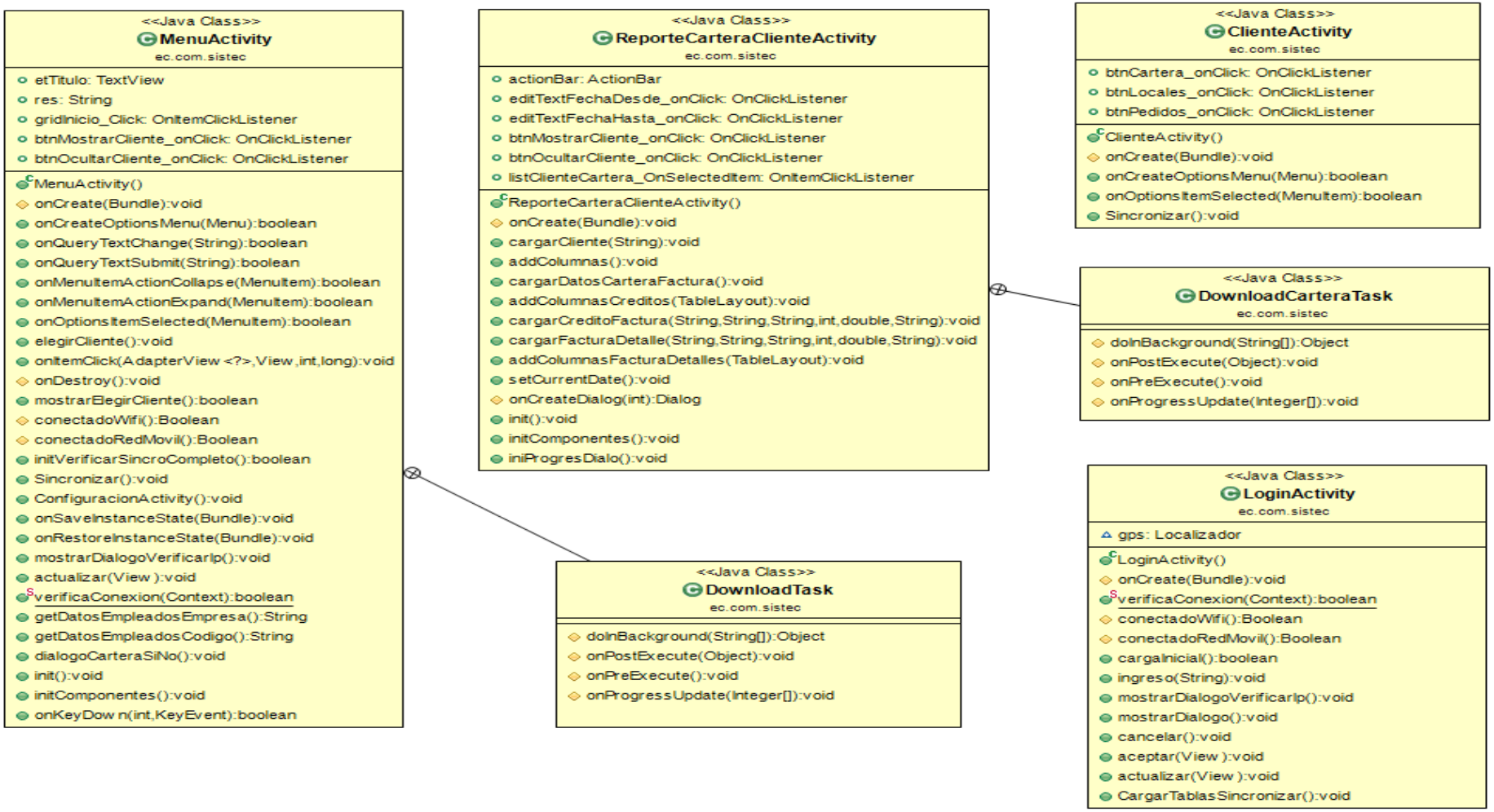


Ilustración 29. Clases que conforman el paquete “ec.com.sistec”**. Fuente:** El autor

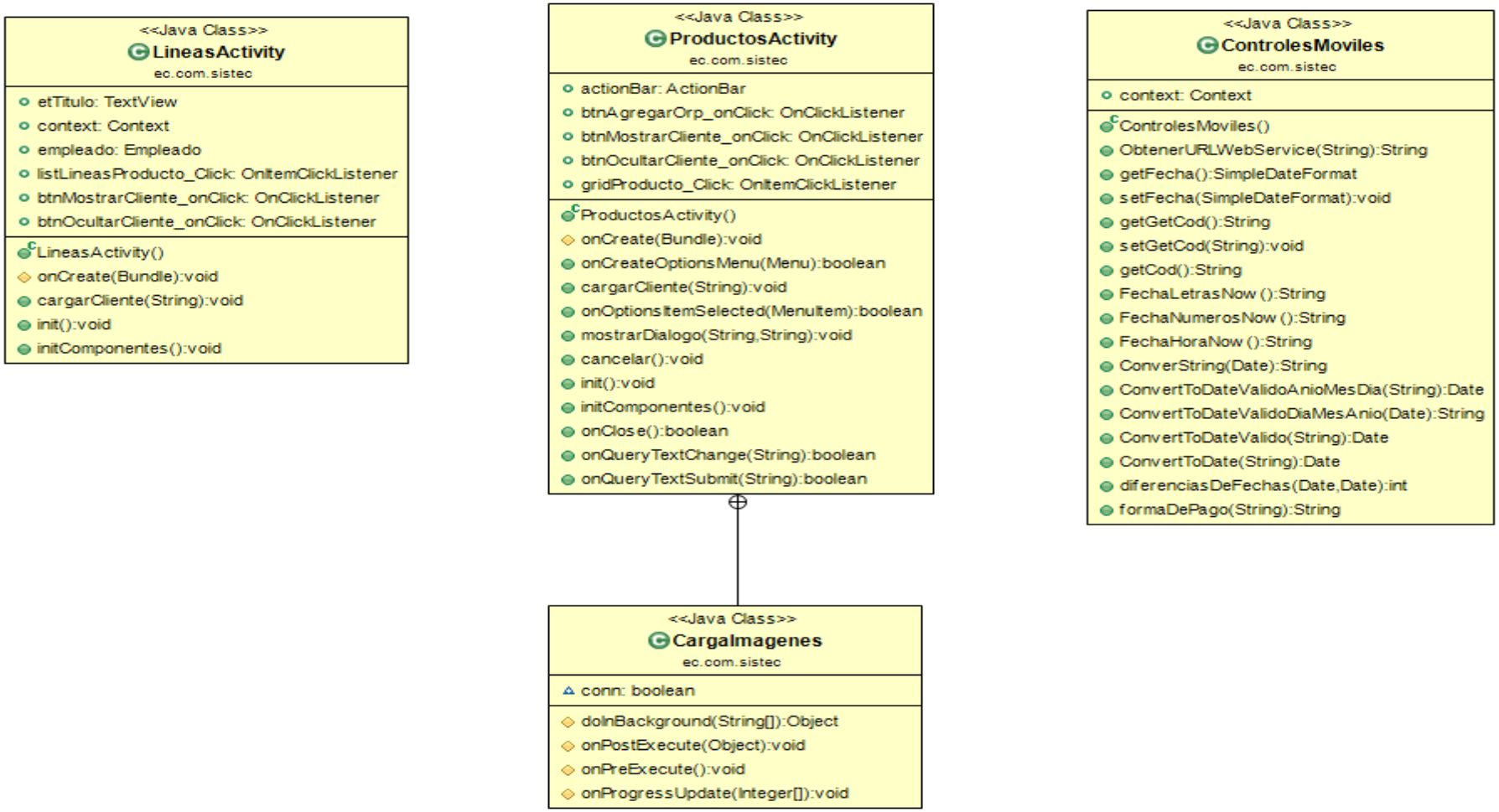


Ilustración 30. Clases que conforman el paquete “ec.com.sistec”. **Fuente:** El autor

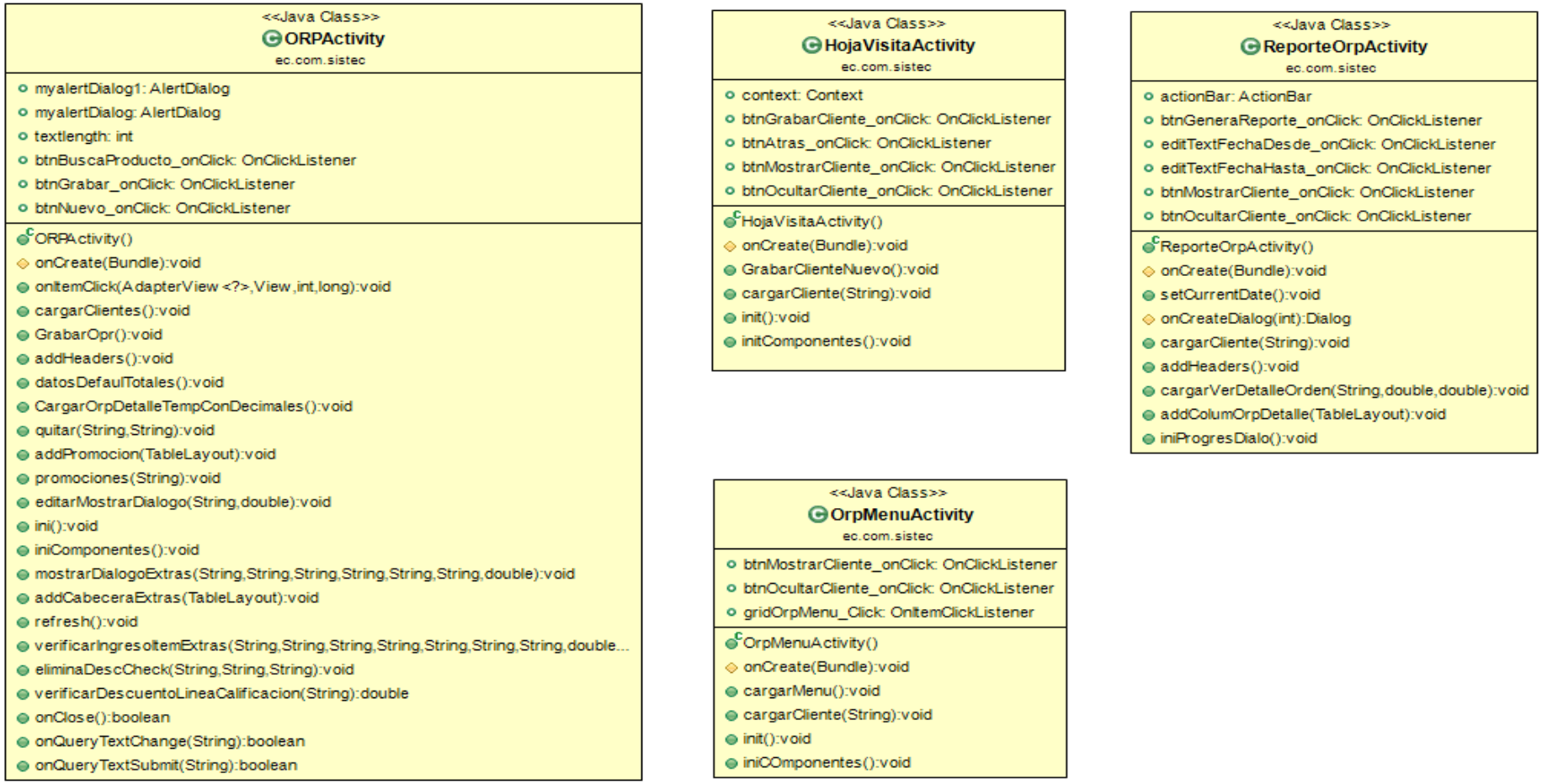


Ilustración 31. Clases que conforman el paquete “ec.com.sistec”. **Fuente:** El autor

El paquete *"ec.com.sistec.controlador*" se encarga de la gestión y tratamiento de datos a través de los métodos CRUD (Create, Read, Update, Delete). El paquete controlador está formado por las siguientes tablas:

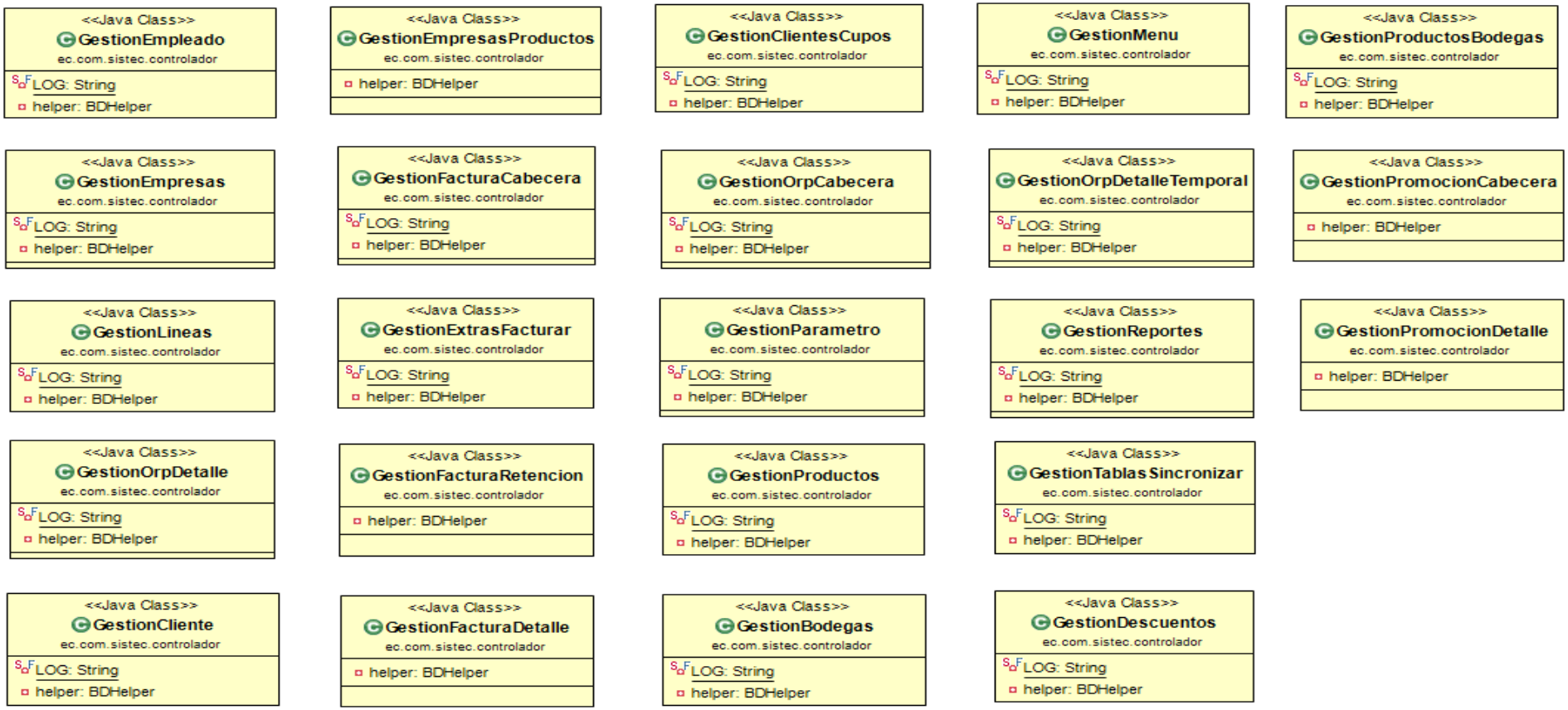


Ilustración 32. Clases que conforman el paquete "ec.com.sistec.controlador". **Fuente:** El autor

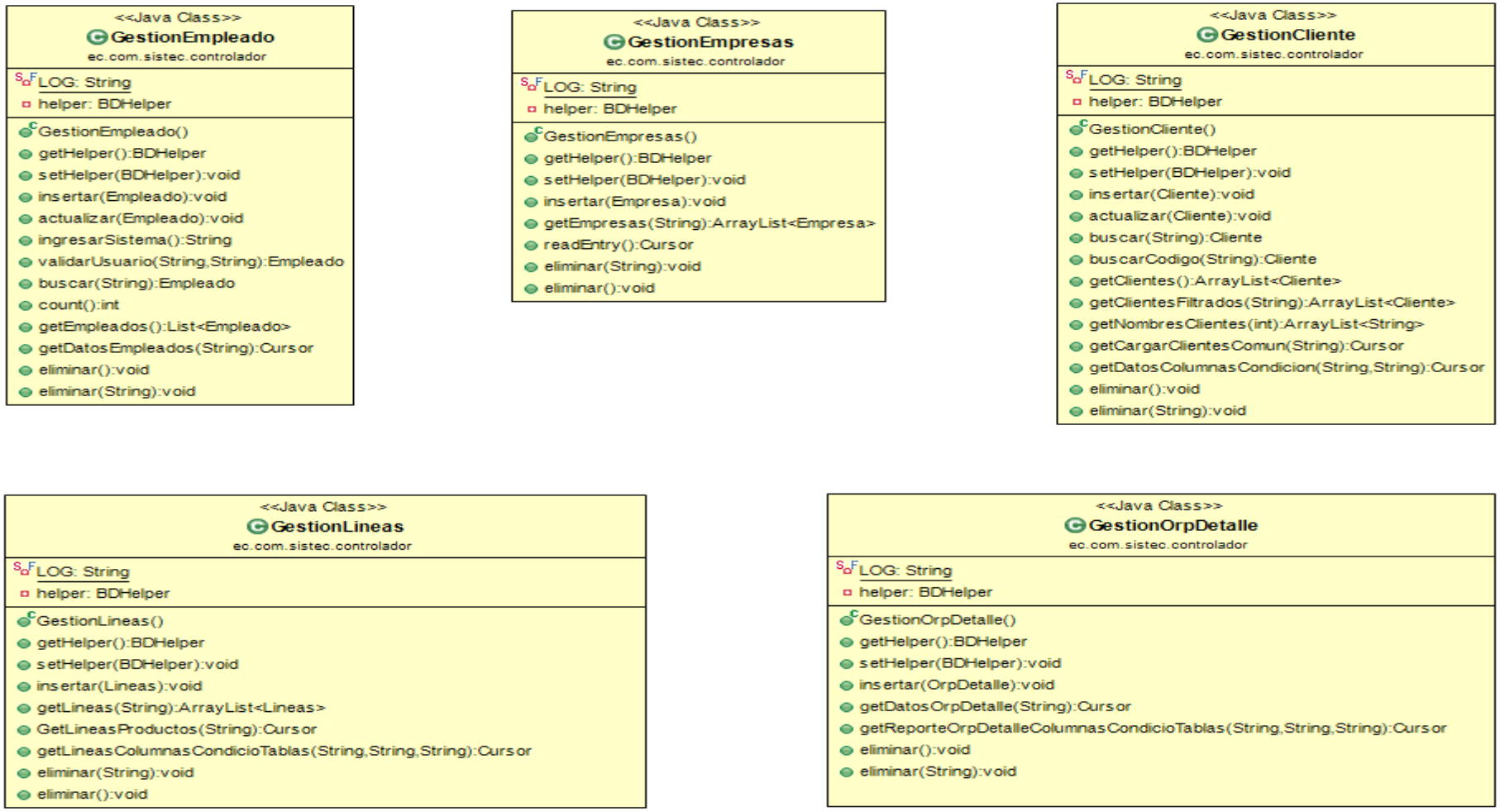


Ilustración 33. Clases que conforman el paquete "ec.com.sistec.controlador". **Fuente:** El autor

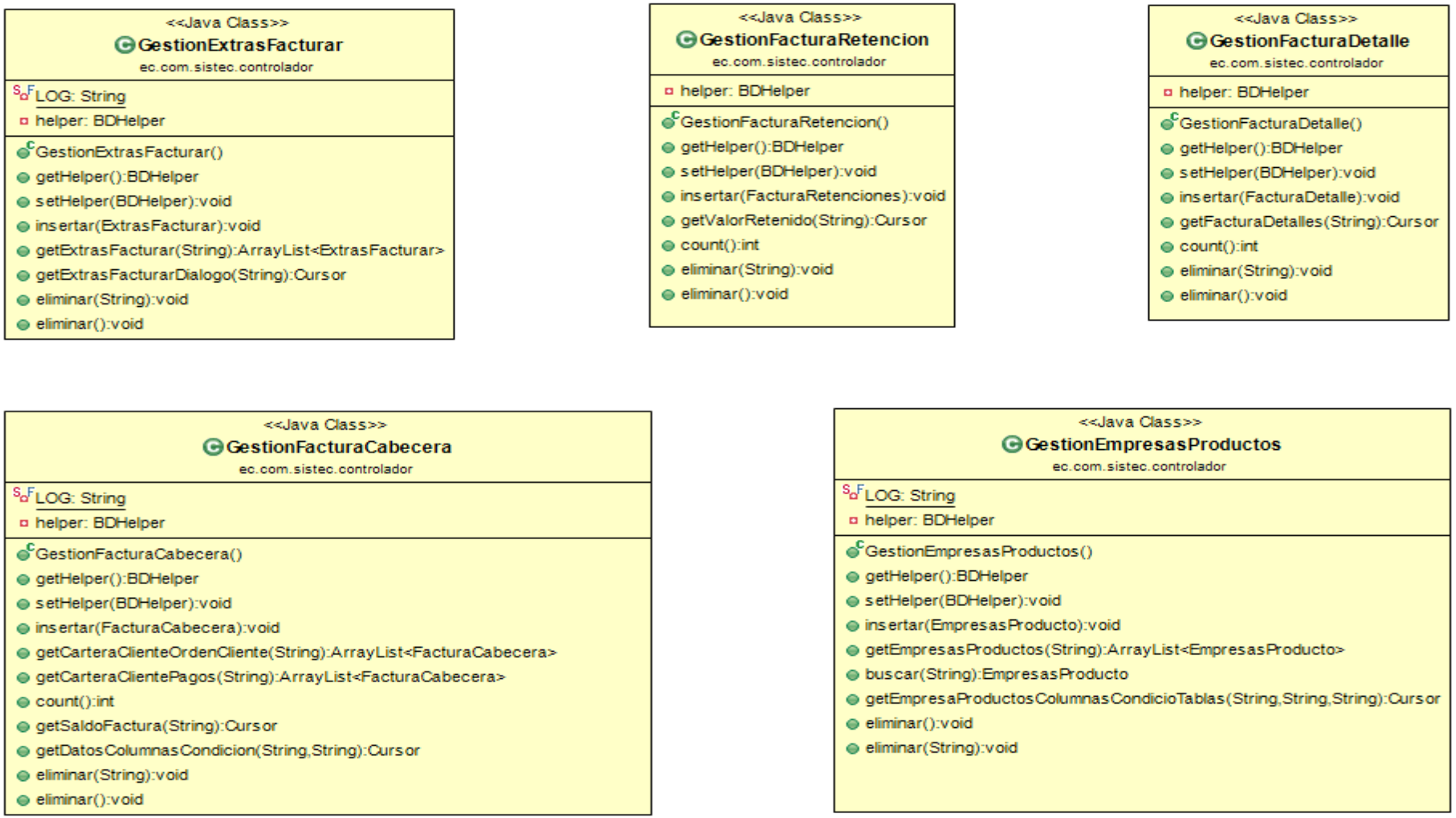


Ilustración 34. Clases que conforman el paquete "ec.com.sistec.controlador"**. Fuente:** El autor

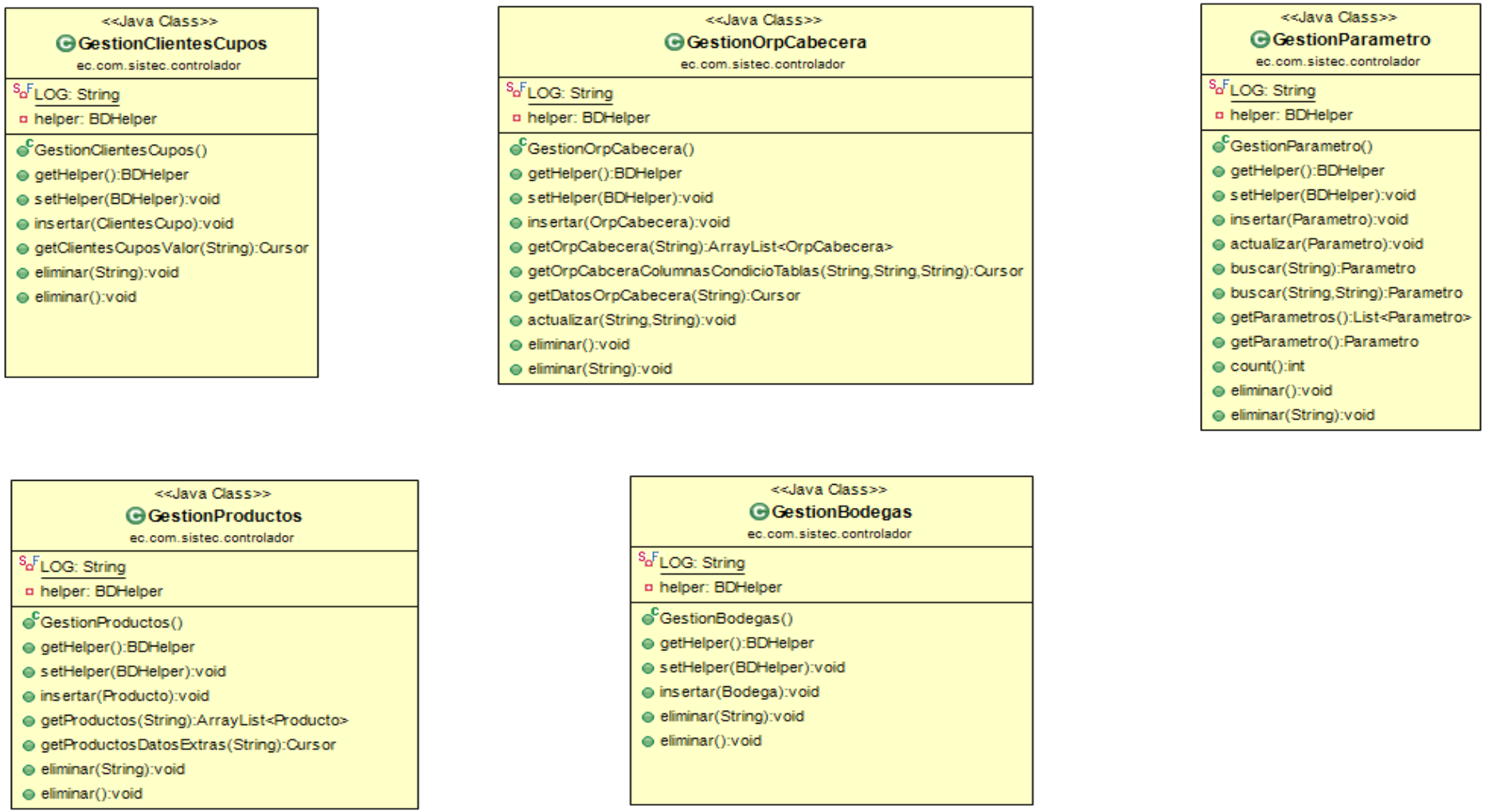


Ilustración 35. Clases que conforman el paquete “ec.com.sistec.controlador”. **Fuente:** El autor

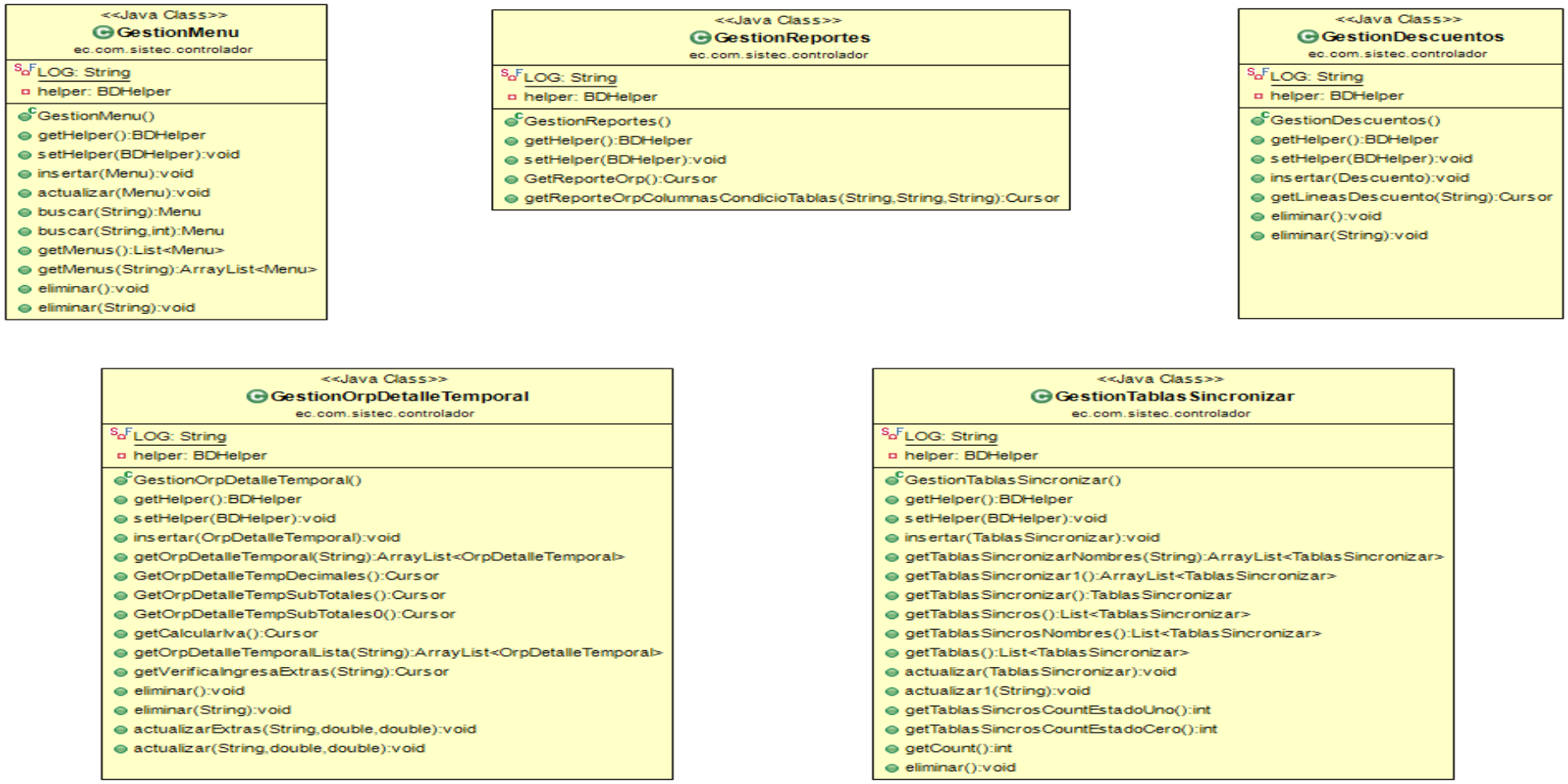


Ilustración 36. Clases que conforman el paquete "ec.com.sistec.controlador". **Fuente:** El autor

Ilustración 37. Clases que conforman el paquete "ec.com.sistec.controlador". **Fuente:** El autor

El paquete *"ec.com.sistec. modelo"* contiene las entidades de las tablas, los mismos que se conectan a los orígenes de datos mediante métodos set y get. Las tablas que conforman el paquete modelo son:

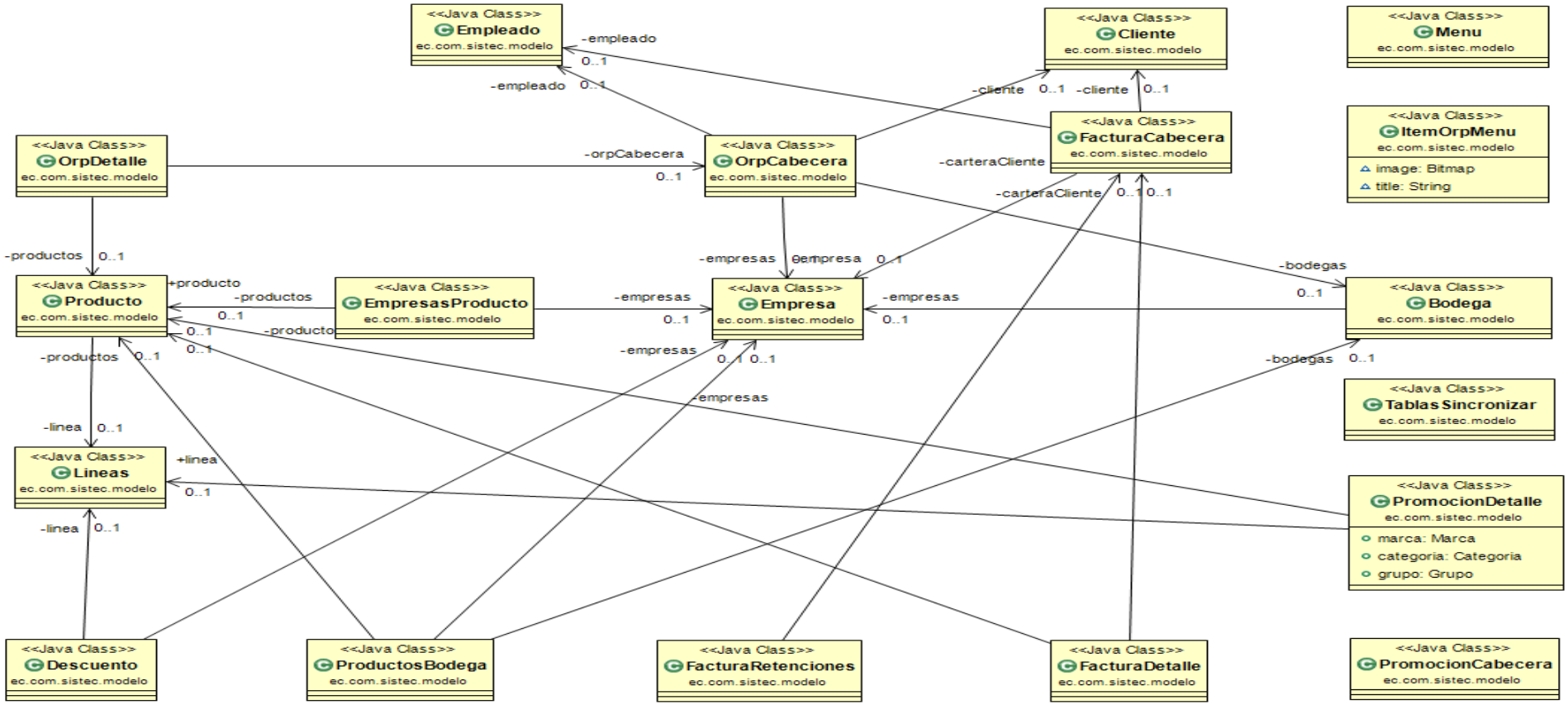


Ilustración 38. Diagrama de Clases del paquete "ec.com.sistec. modelo". **Fuente:** El autor

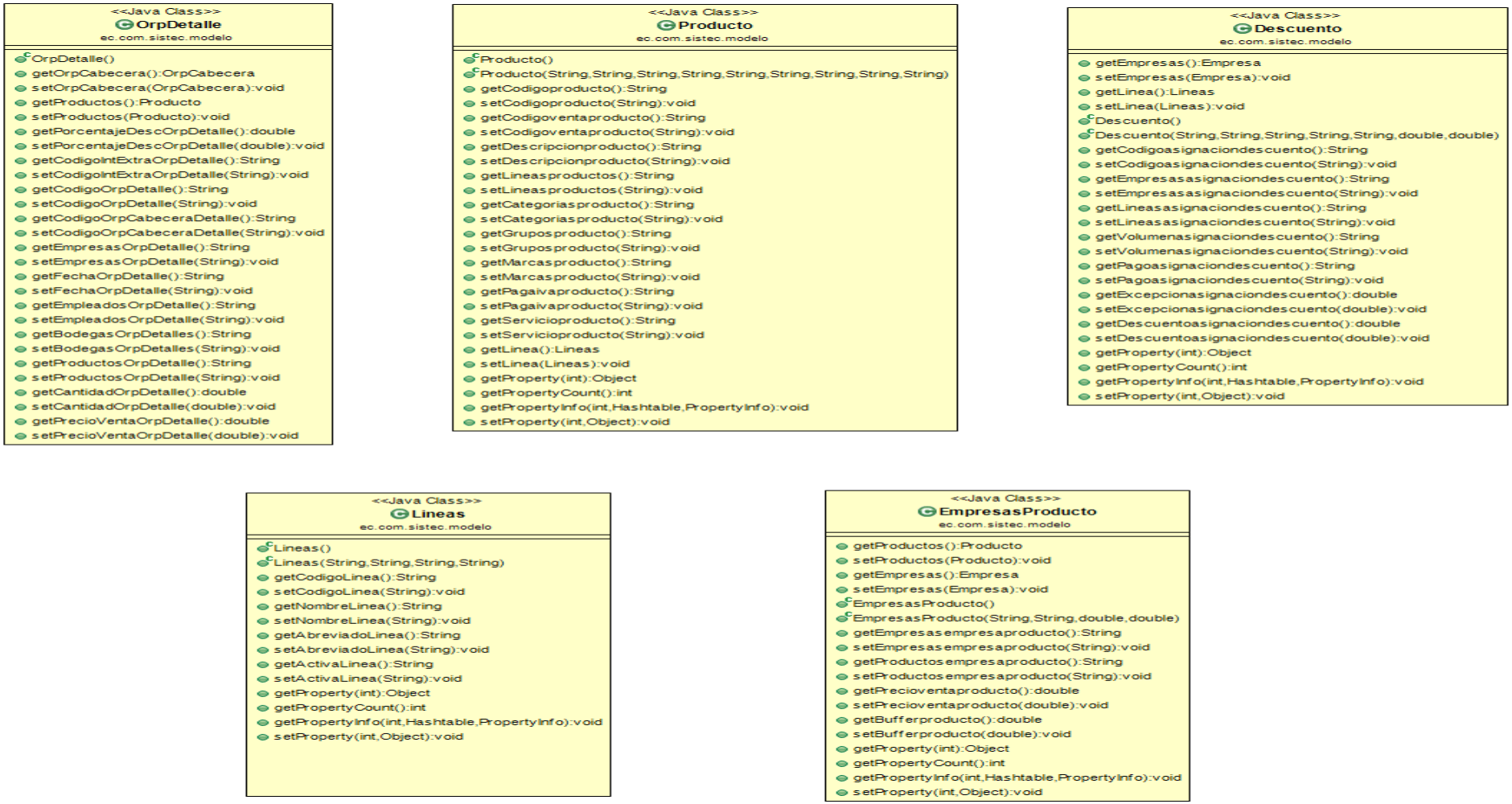


Ilustración 39. Clases que conforman el paquete “ec.com.sistec. modelo”. **Fuente:** El autor

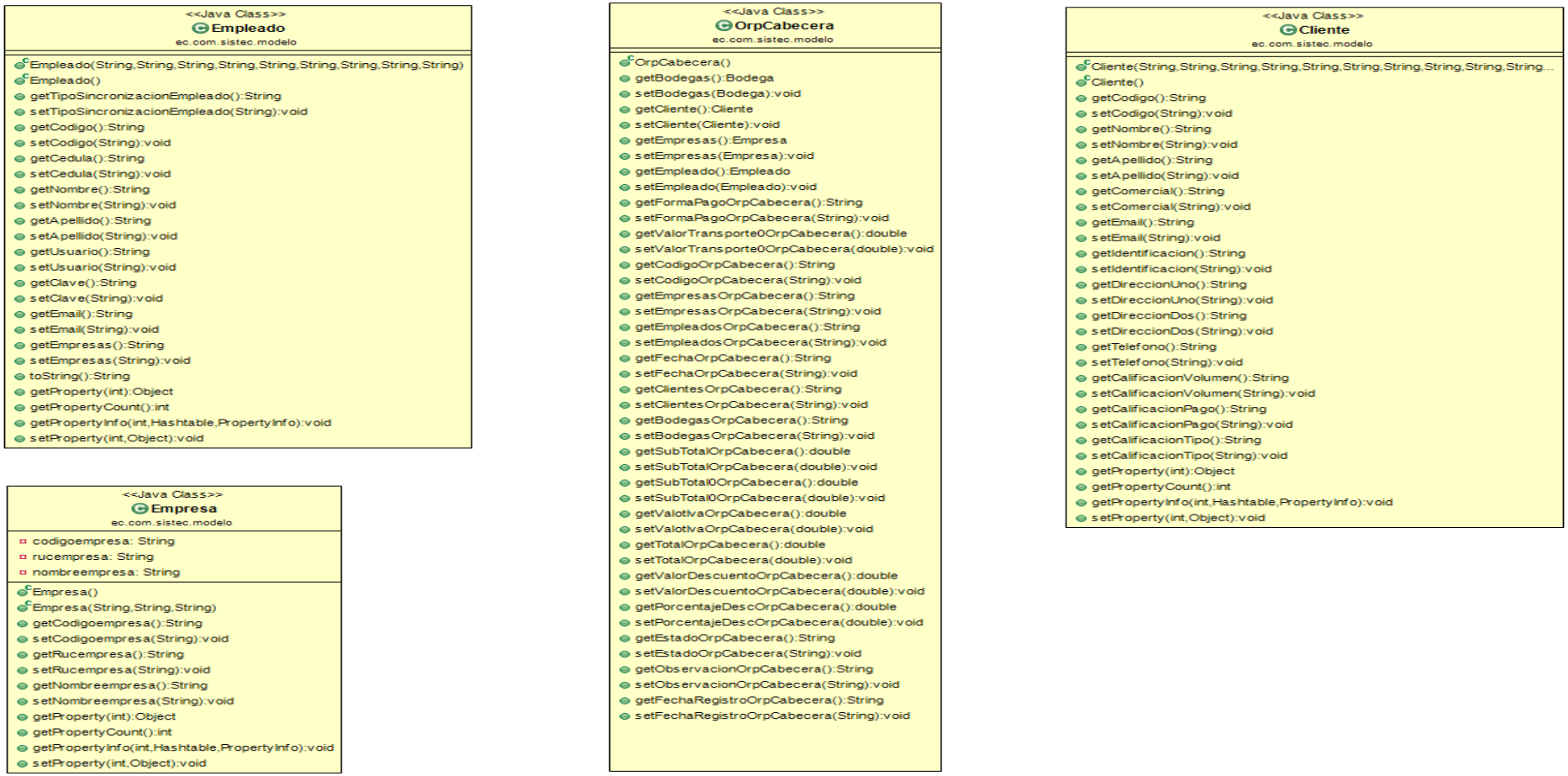
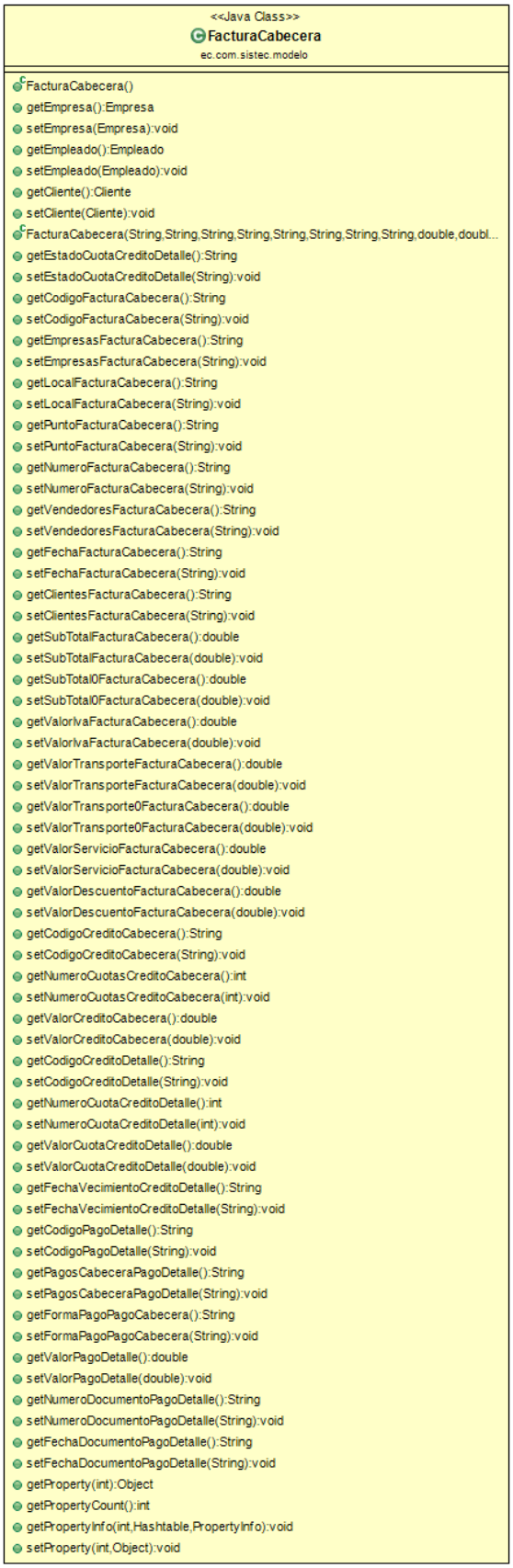
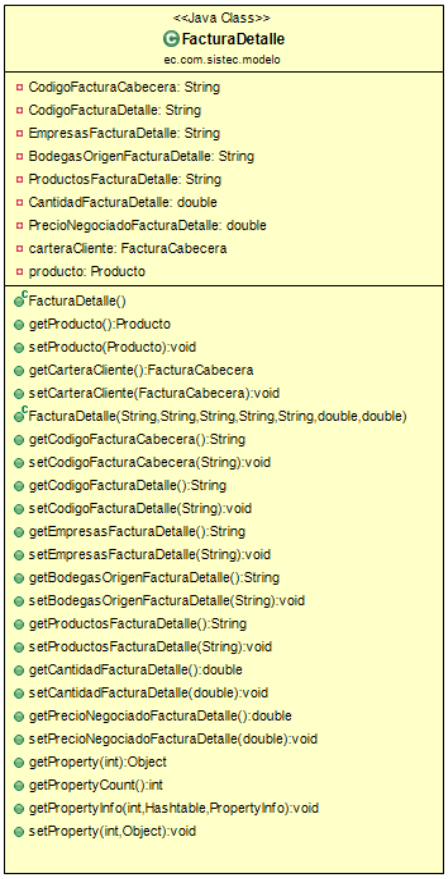


Ilustración 40. Clases que conforman el paquete “ec.com.sistec. modelo”. **Fuente:** El autor





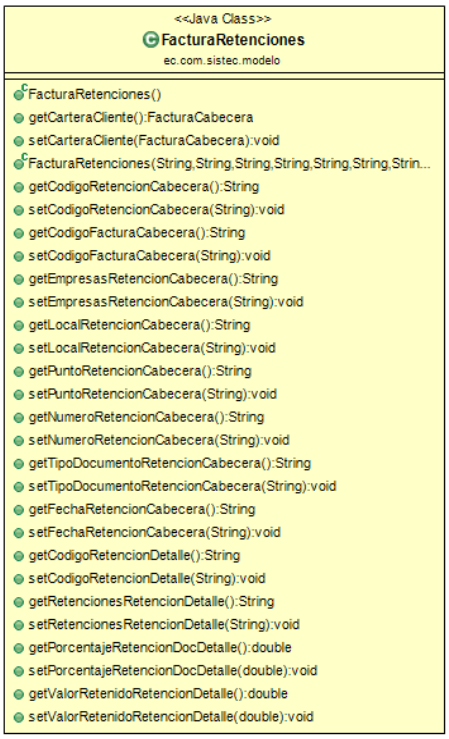


Ilustración 41. Clases que conforman el paquete “ec.com.sistec. modelo”. **Fuente:** El autor

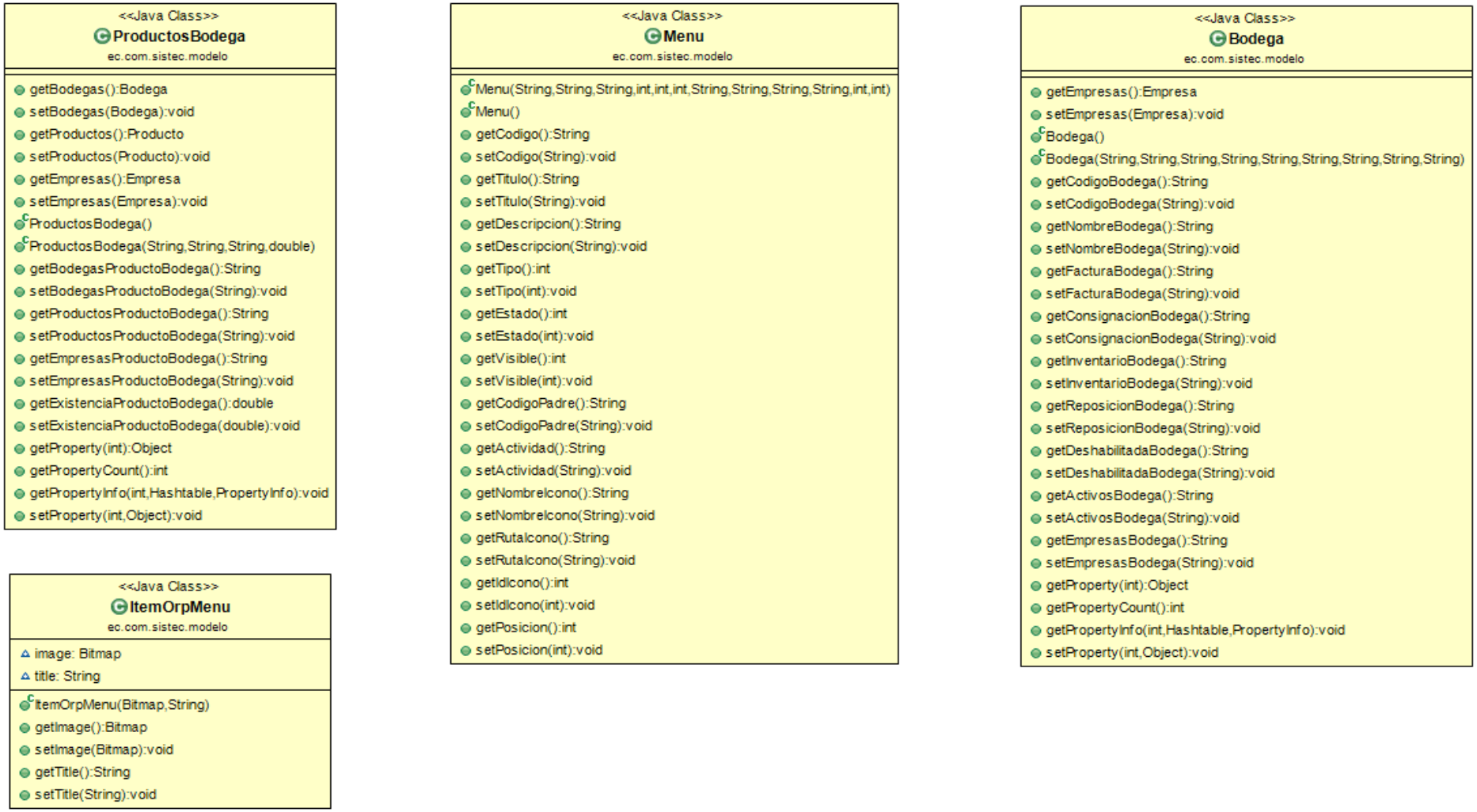


Ilustración 42. Clases que conforman el paquete “ec.com.sistec. modelo”. **Fuente:** El autor

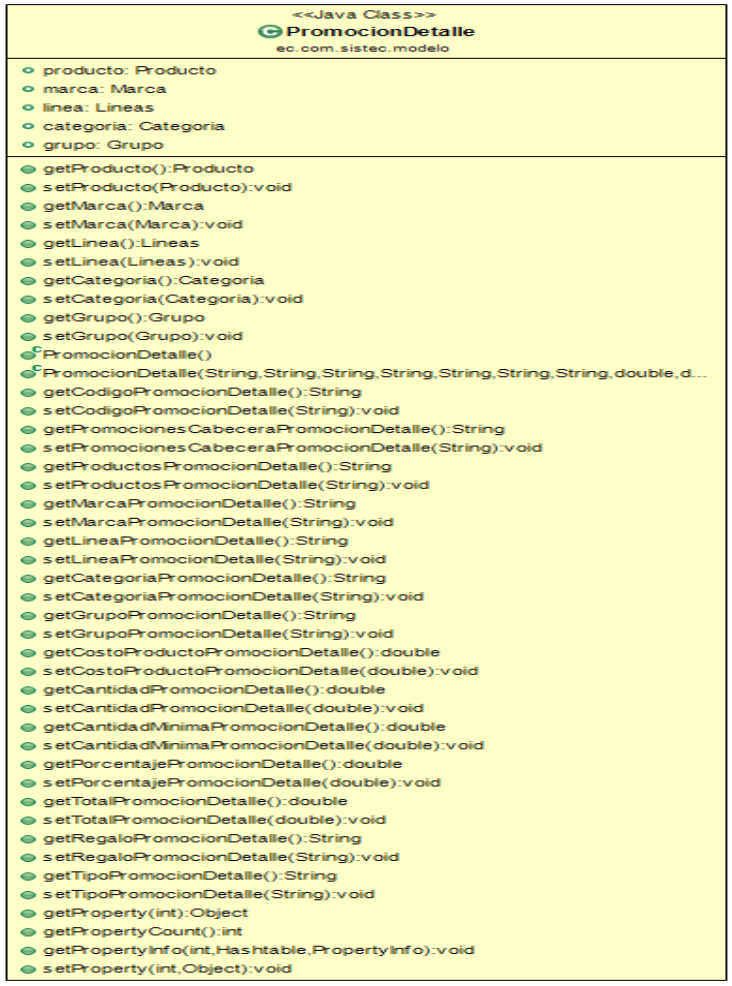


Ilustración 43. Clases que conforman el paquete “ec.com.sistec. modelo”. **Fuente:** El autor

El paquete “ec.com.sistec.origen” contiene el esquema lógico de base de datos SQLite y hace uso de la clase principal de Android llamada *BDHelper*. Esta clase contiene el método *onCrete()* encargado de crear la base de datos y el método *onUpgrade ()* que permite actualizarla. El paquete está compuesto por:

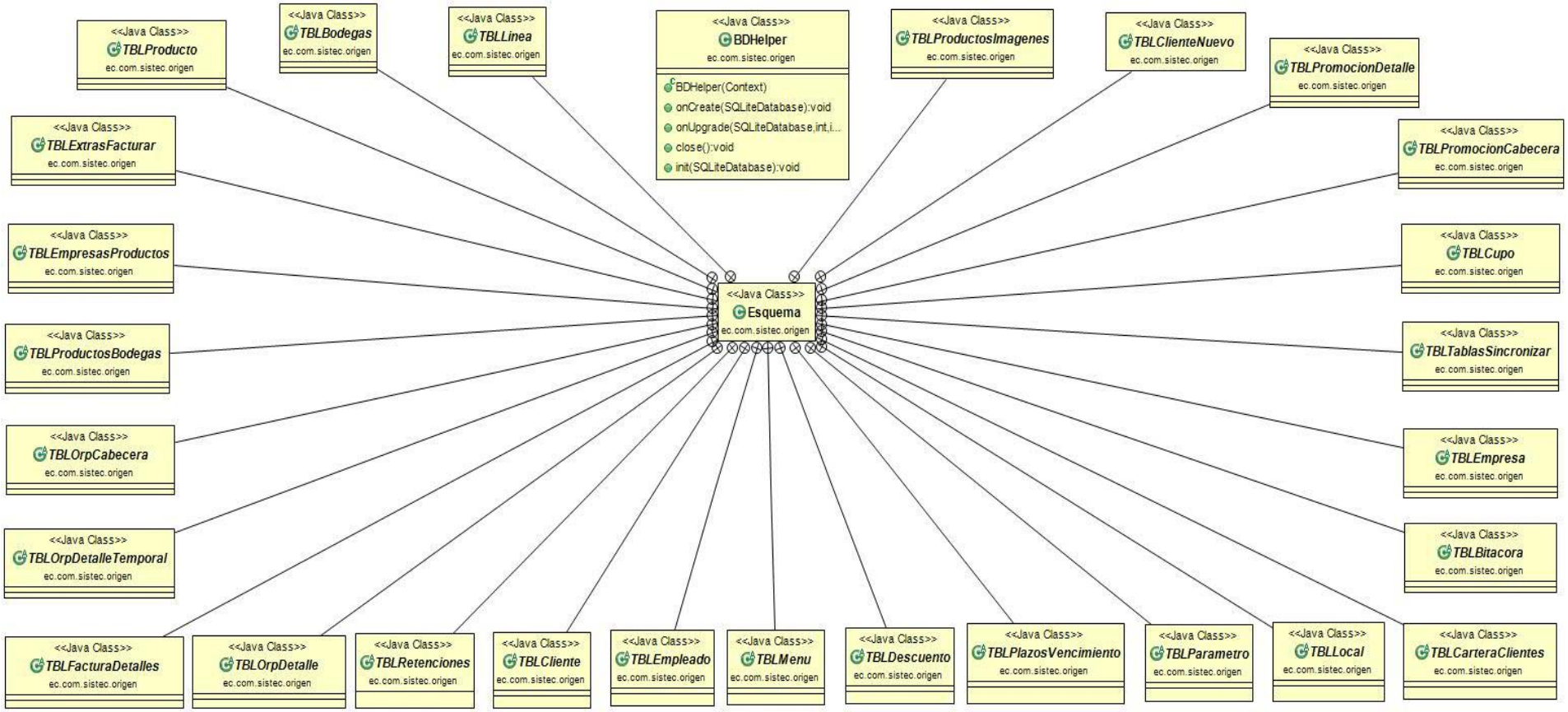


Ilustración 44. Diagrama de Clases del paquete “ec.com.sistec.origen”. **Fuente:** El autor

El paquete "*ec.com.sistec.servicio*" contiene los servicios de la aplicación, es decir las clases que se ejecutan en segundo plano para el envió de datos al servidor central.

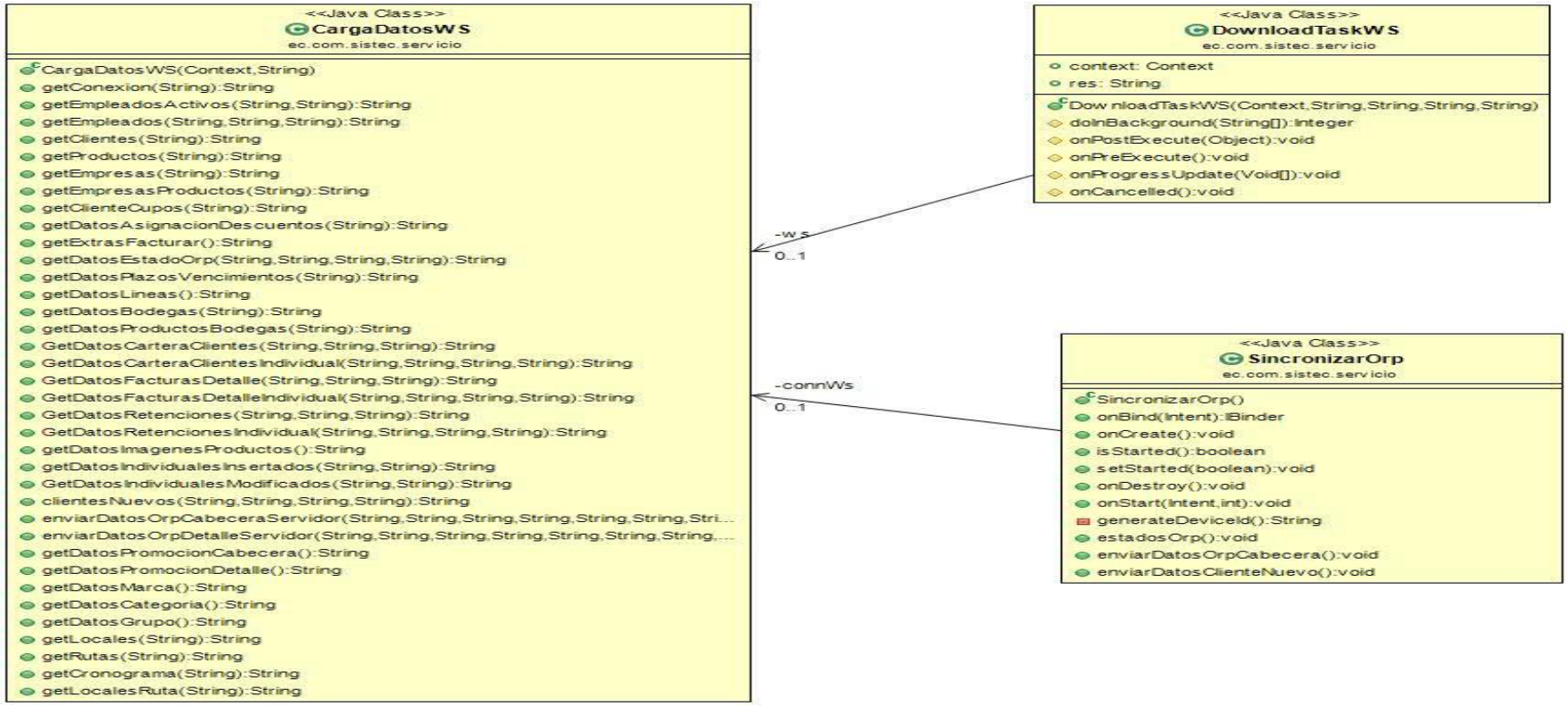


Ilustración 45. Diagrama de Clases del paquete "ec.com.sistec.servicio". **Fuente:** El autor

El paquete "*ec.com.sistec.utilidades*" proporciona una interfaz común al modelo de datos, todos los controladores accederán a los datos a través de los diferentes adaptadores. Ejemplo: AdarptadorProducto lista los productos de acuerdo a una categorización, AdaptadorCarteraCliente lista el detalle de la cartera de un determinado cliente, etc.

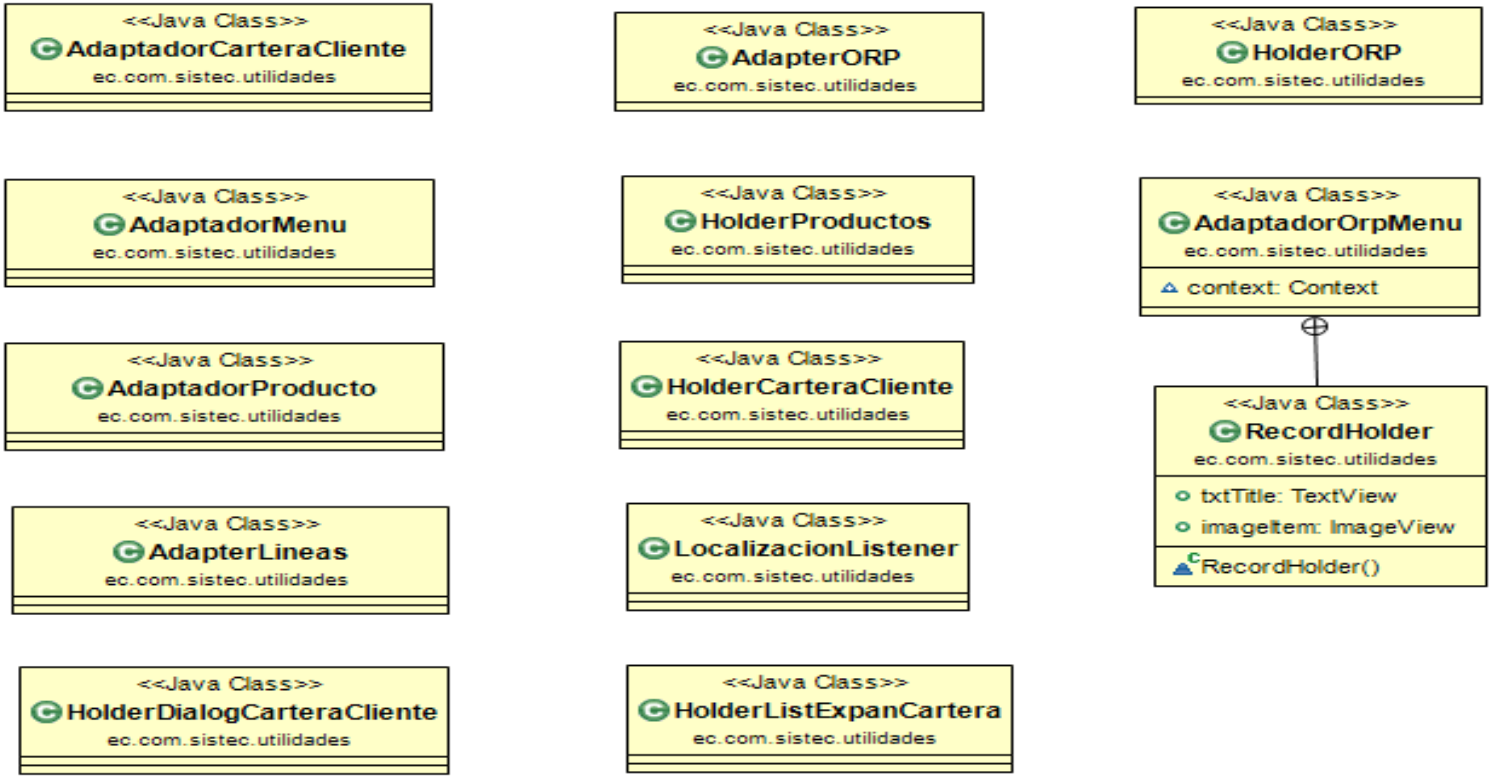


Ilustración 46. Clases que conforman el paquete "ec.com.sistec.utilidades". **Fuente:** El autor

Ilustración 47. Clases que conforman el paquete "ec.com.sistec.utilidades". **Fuente:** El autor

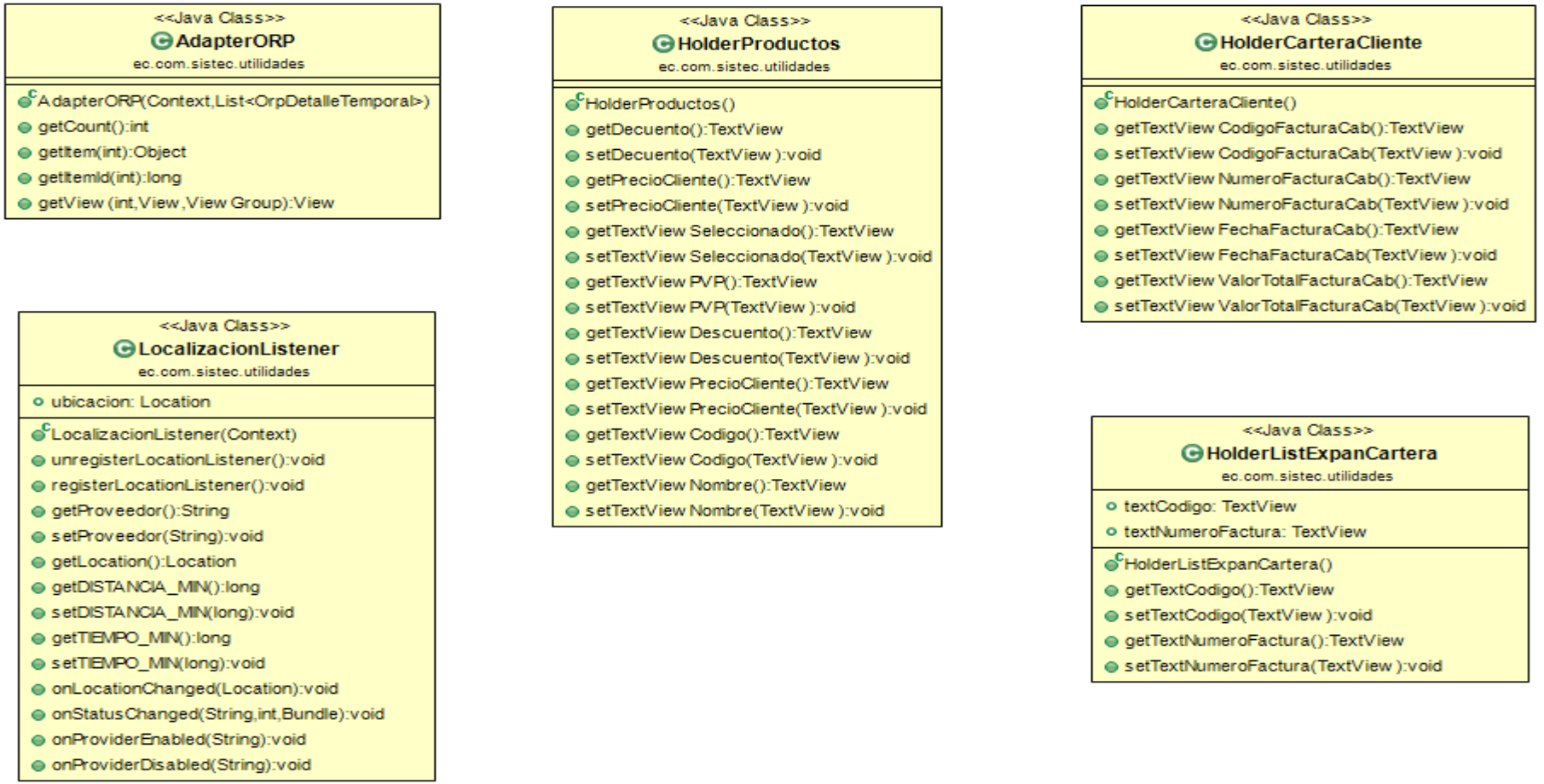


Ilustración 48. Clases que conforman el paquete “ec.com.sistec.utilidades”. **Fuente:** El autor

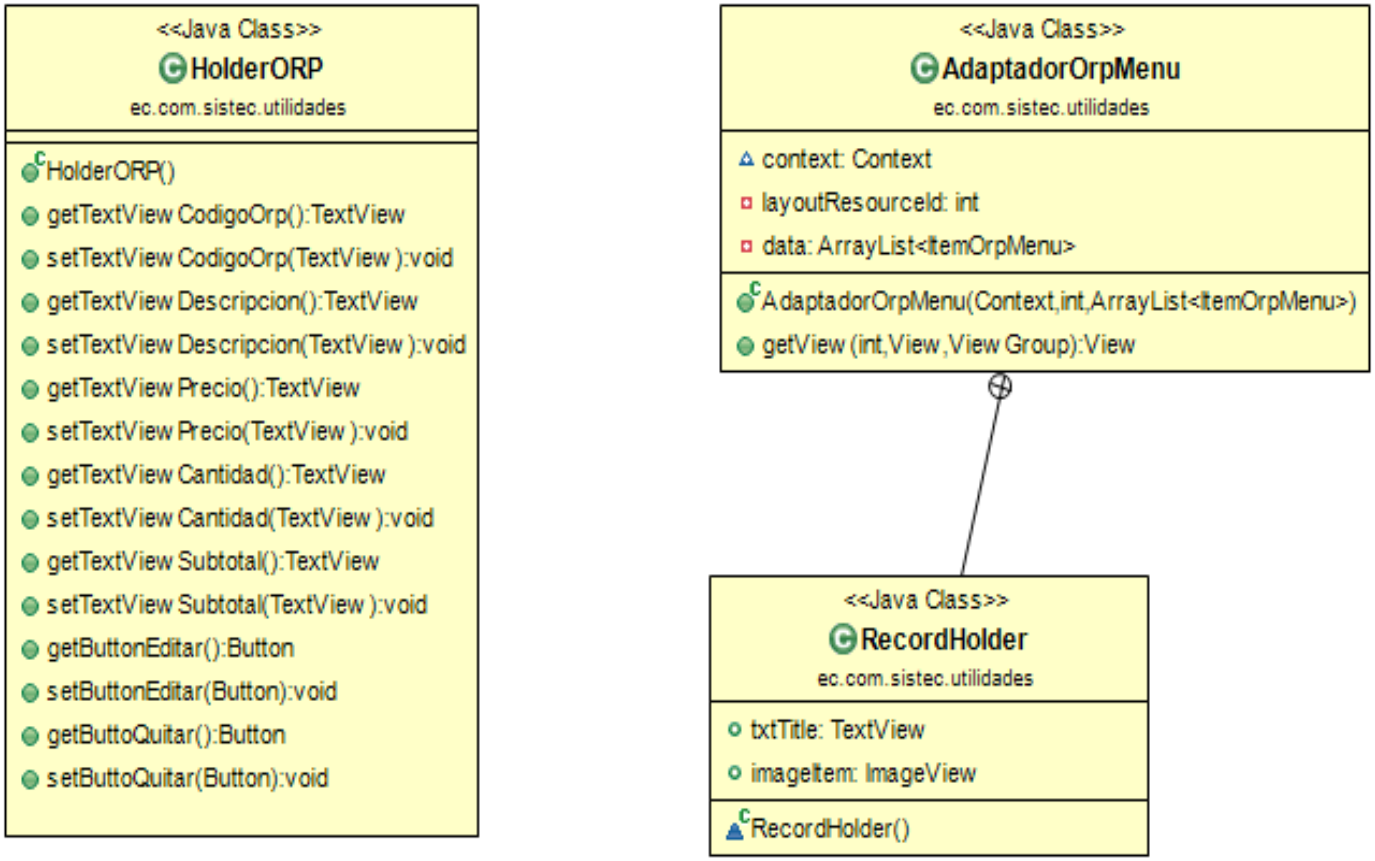


Ilustración 49. Clases que conforman el paquete “ec.com.sistec.utilidades”. **Fuente:** El autor

Los paquetes relacionados *"ec.com.sistec. modelo"* con *"ec.com.sistec.controlador*" contiene las siguientes clases:

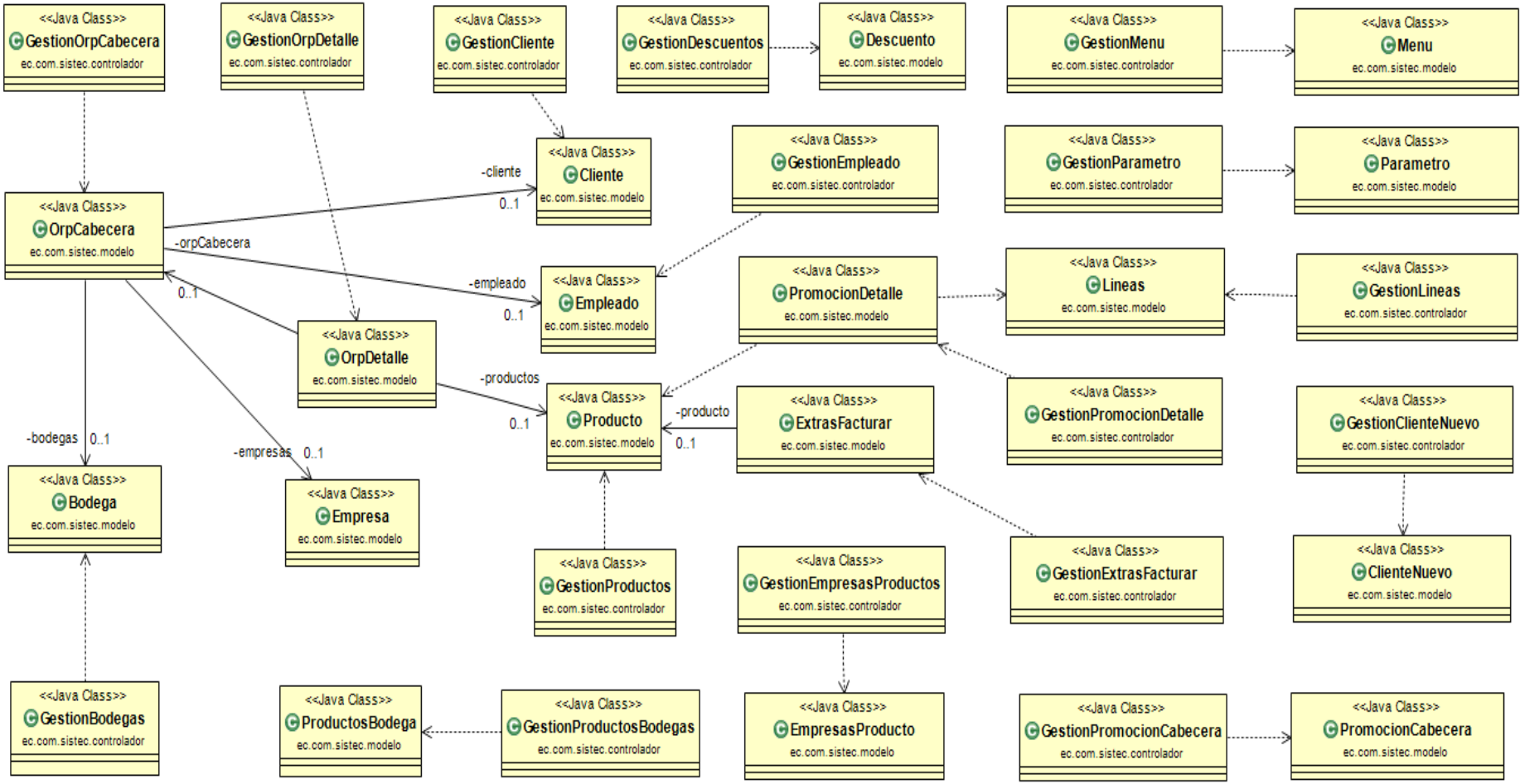


Ilustración 50. Diagrama de Clases del paquete "ec.com.sistec. modelo" con "ec.com.sistec.controlador"**. Fuente:** El autor

Los paquetes relacionados *"ec.com.sistec.origen"* con *"ec.com.sistec.controlador"* contienen las siguientes clases:

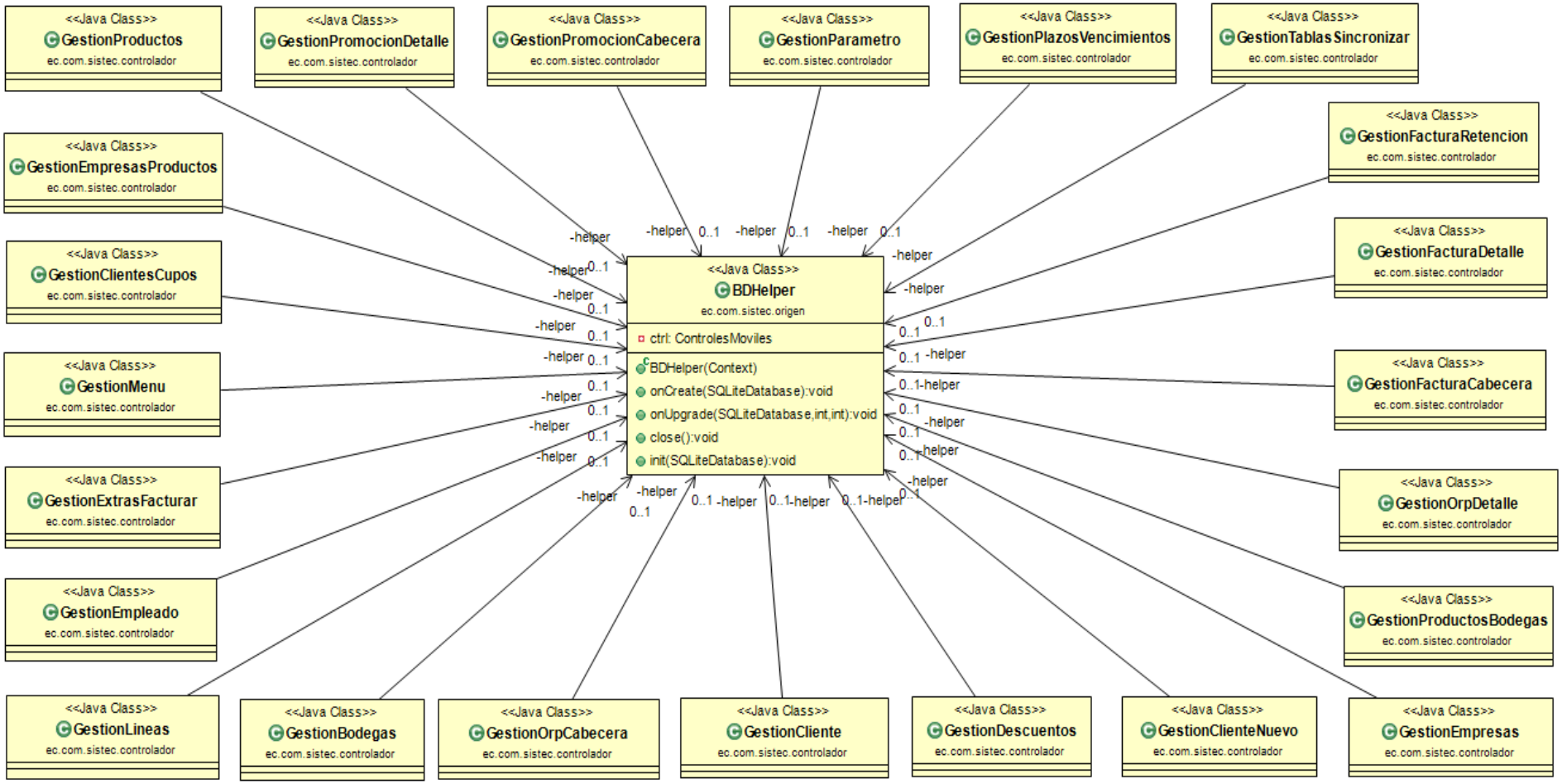


Ilustración 51. Diagrama de Clases del paquete "ec.com.sistec.origen" con "ec.com.sistec.controlador"**. Fuente:** El autor

Los paquetes relacionados *"ec.com.sistec"* con *"ec.com.sistec.controlador"* contienen las siguientes clases:

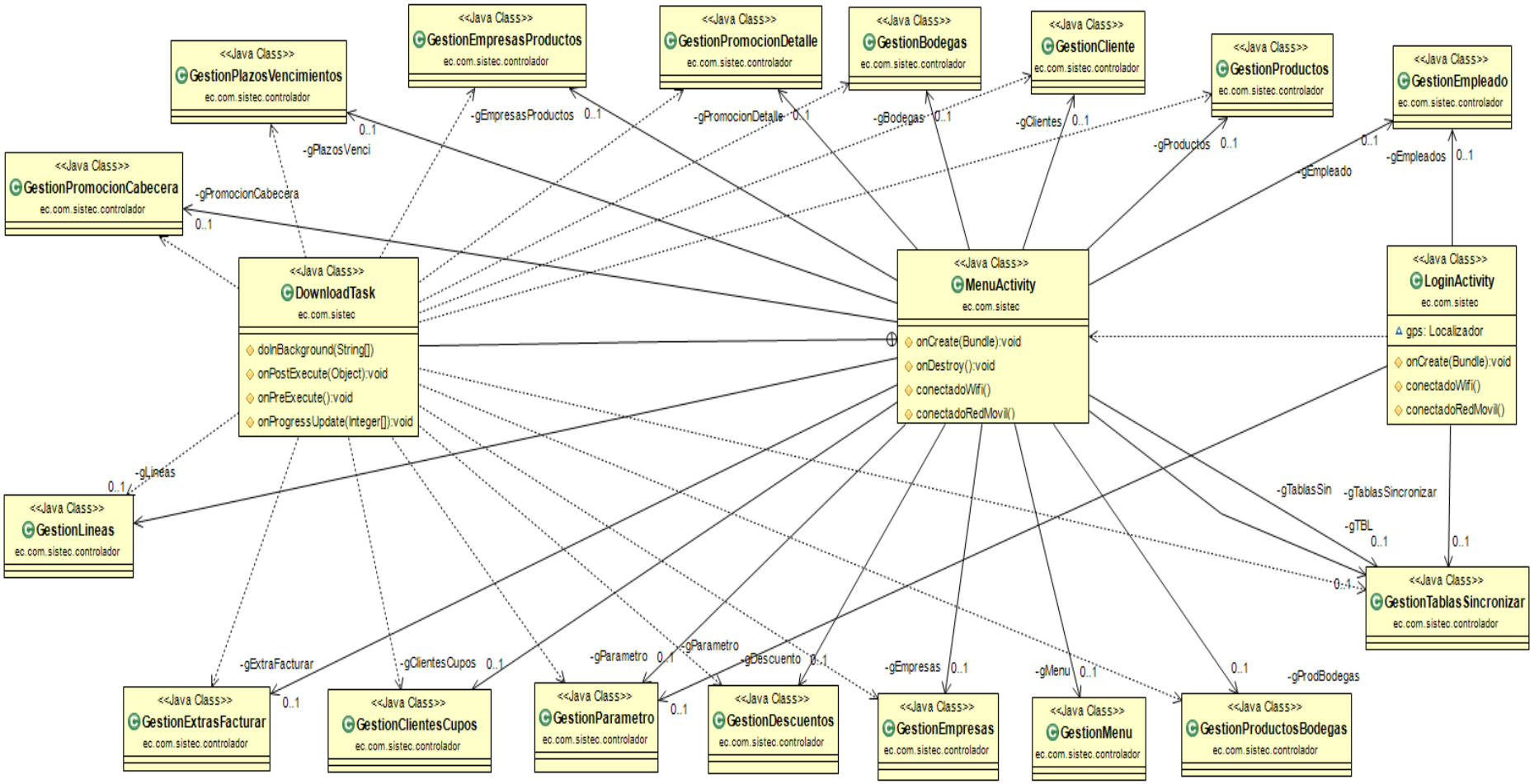


Ilustración 52. Diagrama de Clases del paquete "ec.com.sistec" con "ec.com.sistec.controlador" .**Fuente:** El autor

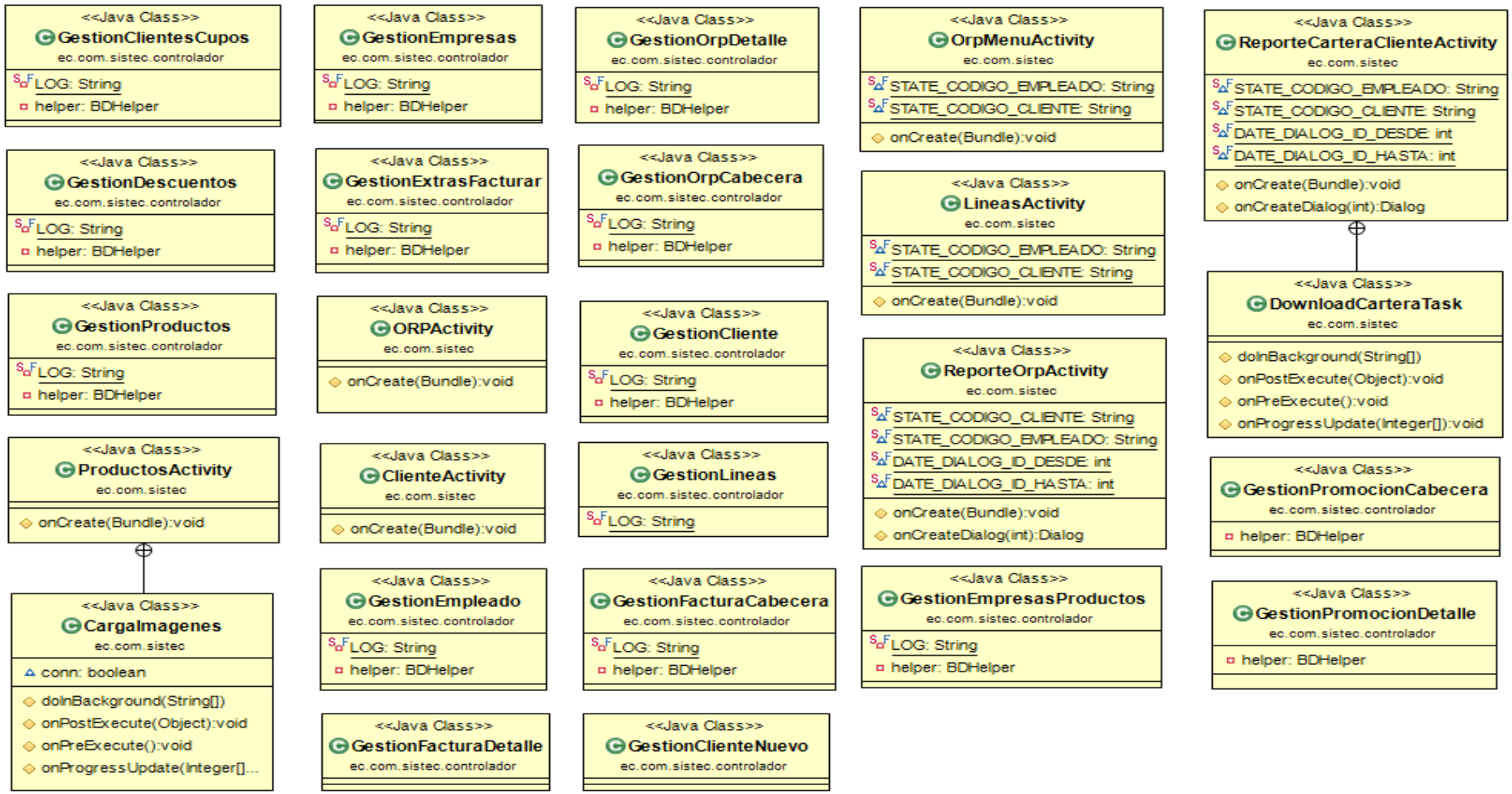


Ilustración 53. Clases que contiene el paquete "ec.com.sistec" con "ec.com.sistec.controlador". **Fuente:** El autor

Los paquetes relacionados *"ec.com.sistec.servicio"* con "*ec.com.sistec.controlador*" contienen las siguientes clases:

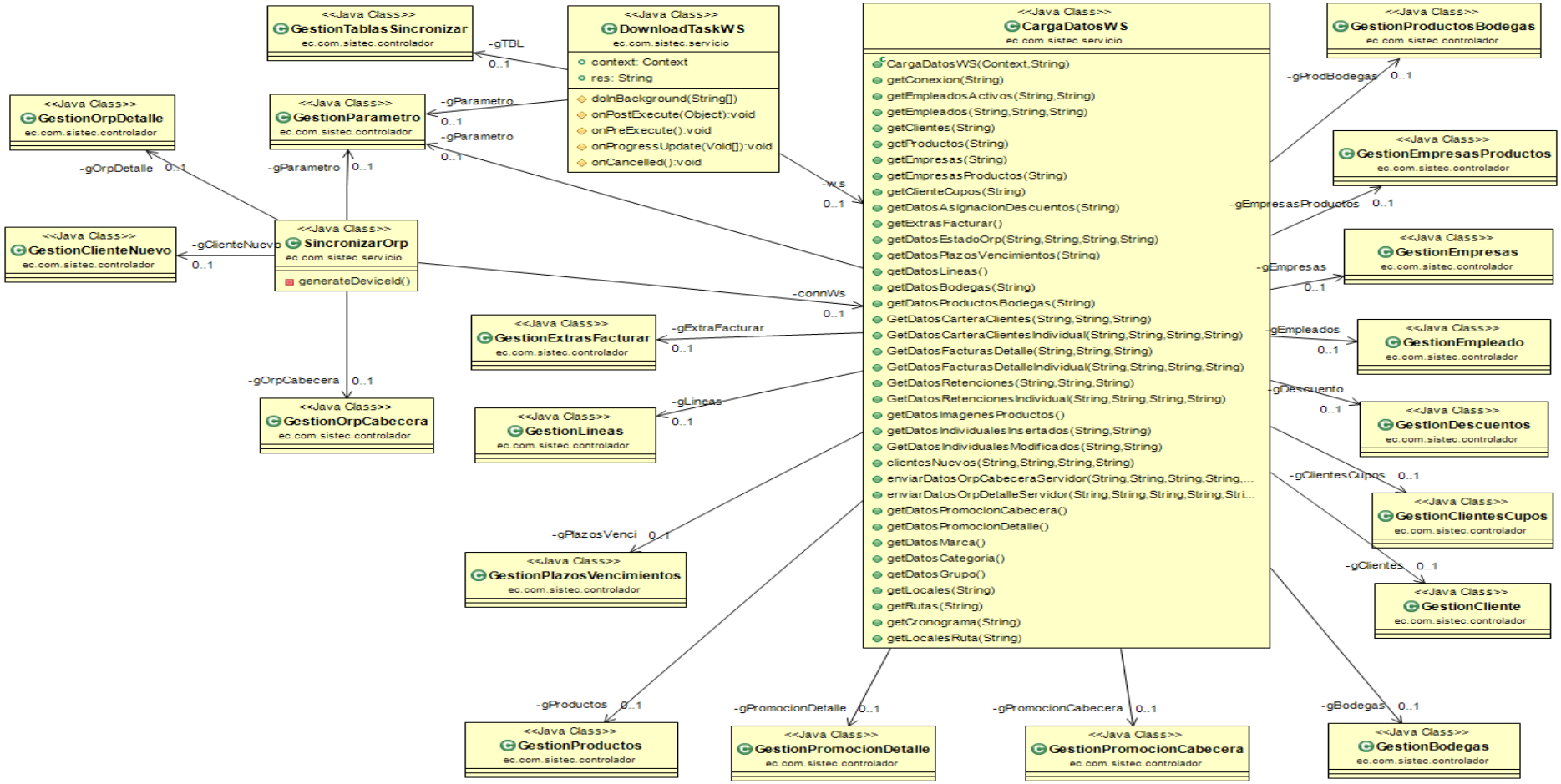


Ilustración 54. Diagrama de Clases del paquete "ec.com.sistec.servicio" con "ec.com.sistec.controlador". **Fuente:** El autor

Los paquetes relacionados *"ec.com.sistec"* con *"ec.com.sistec.modelo"* contienen las siguientes clases:

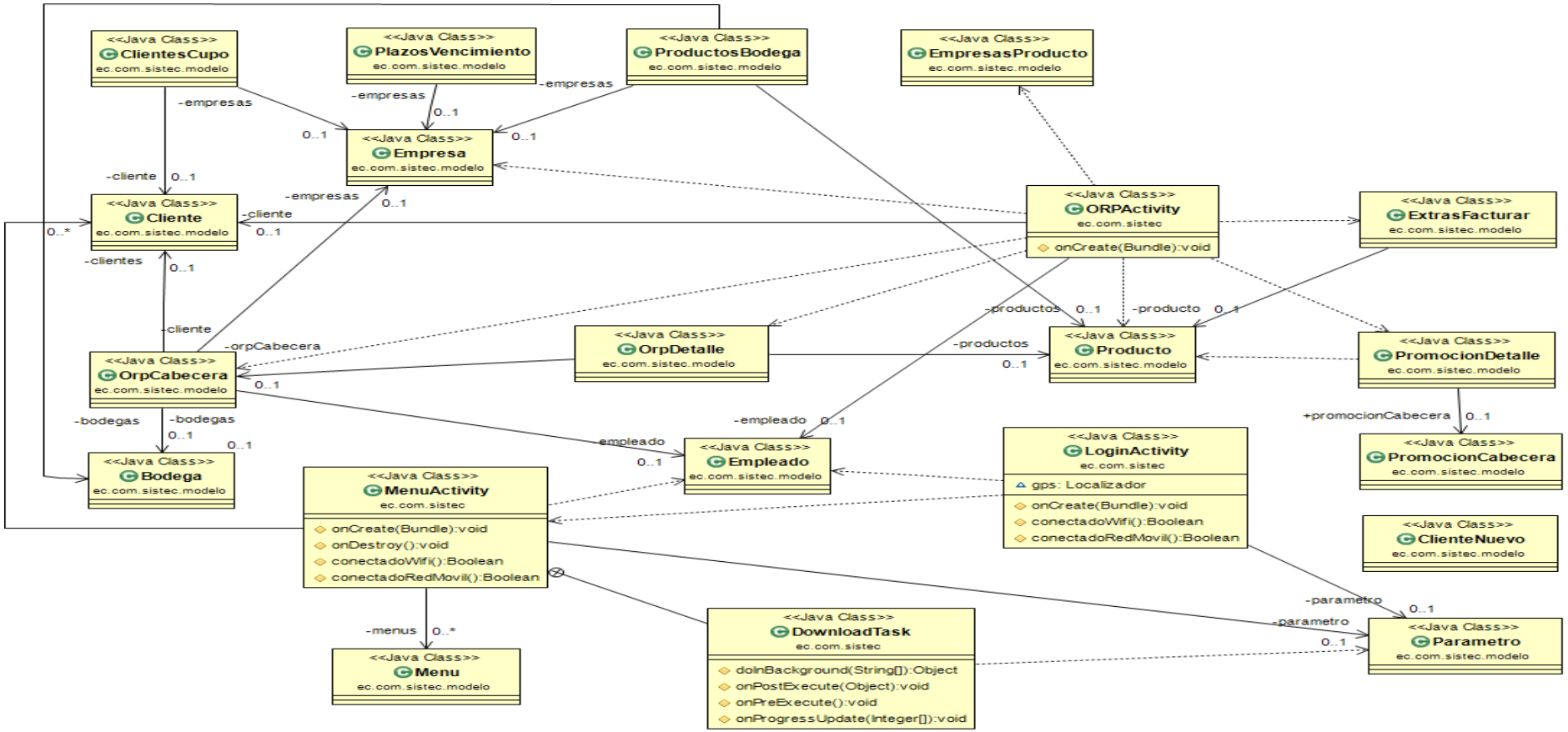


Ilustración 55. Diagrama de Clases del paquete "ec.com.sistec" con "ec.com.sistec.modelo". **Fuente:** El autor

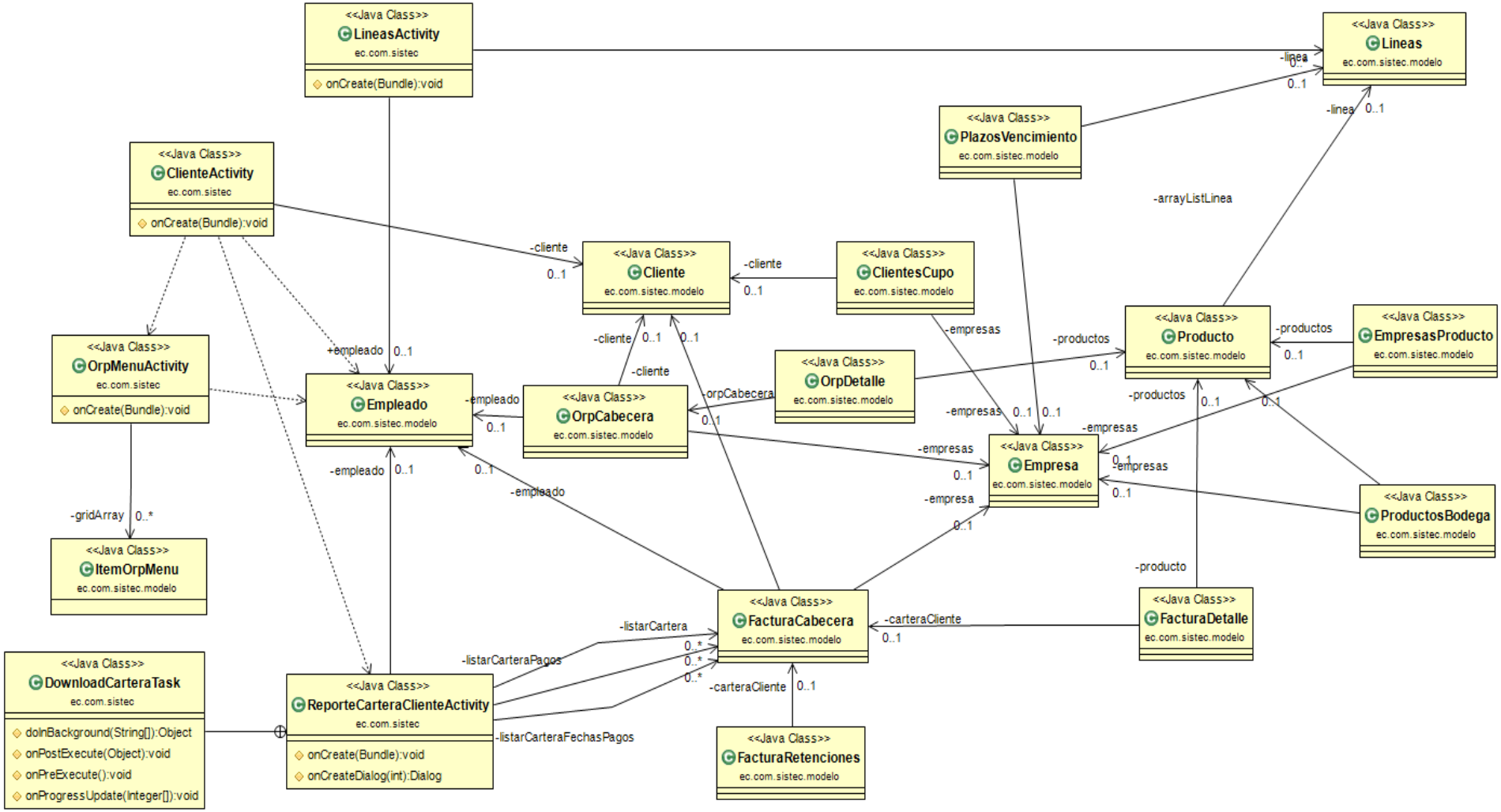


Ilustración 56. Diagrama de Clases del paquete “ec.com.sistec” con “ec.com.sistec.modelo”. **Fuente:** El autor

Los paquetes relacionados *"ec.com.sistec.servicio."* con "ec.com.sistec.modelo" contienen las siguientes clases:

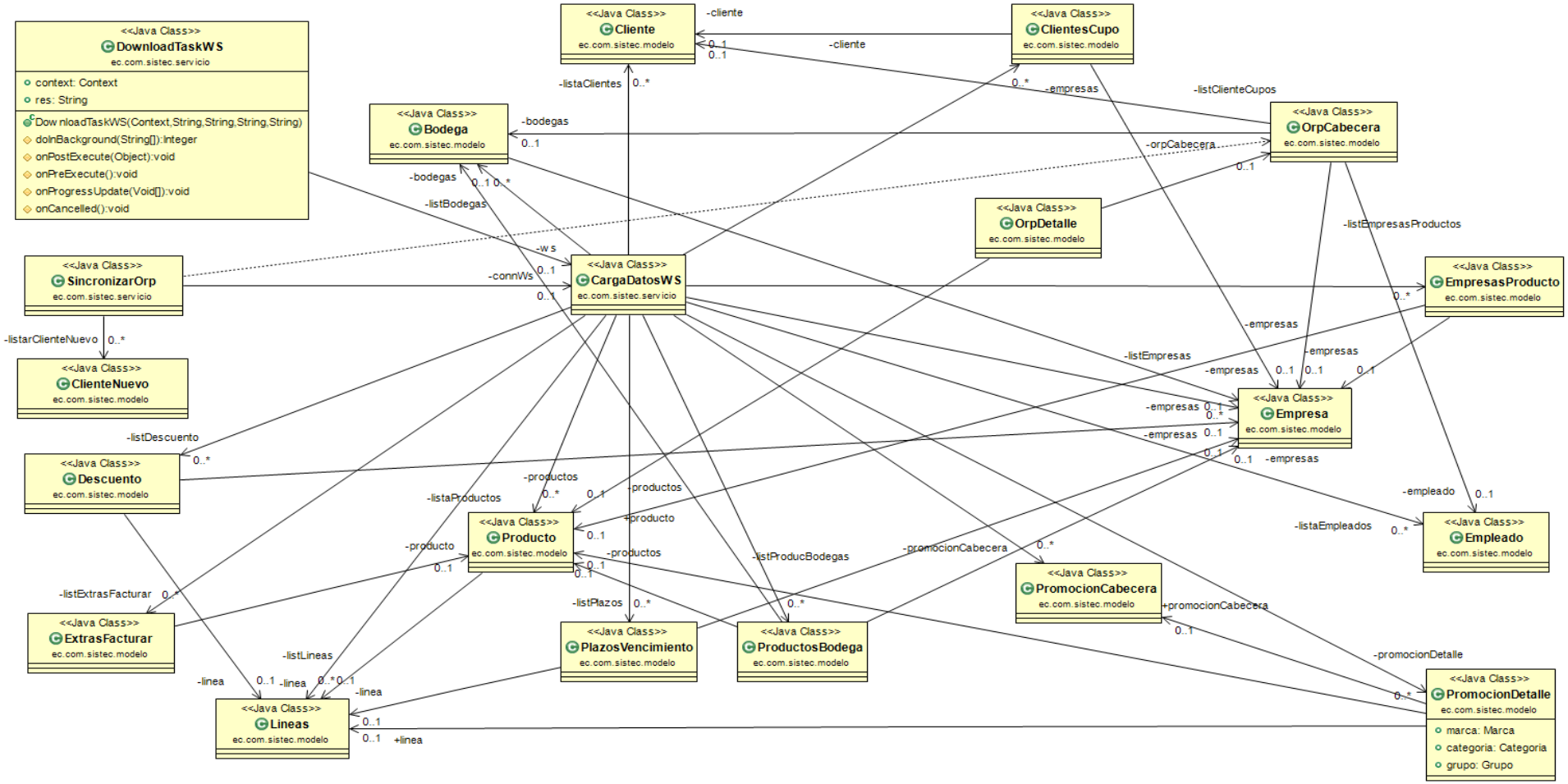


Ilustración 57. Diagrama de Clases del paquete "ec.com.sistec.servicio." con "ec.com.sistec.modelo". **Fuente:** El autor

3.3 Arquitectura lógica y física

3.3.1 Esquema Lógico

El esquema lógico introduce un concepto importante denominado **capas,** son las encargadas de la división lógica de los componenetes, es decir su función es la asignación de tareas y la separación de responsabilidades. La ubicación adecuada de cada componente lógico aportará mayor rendimiento, reutilización de código y facilidad de programación. A continuación se presenta el esquema lógico de la aplicación WISE.

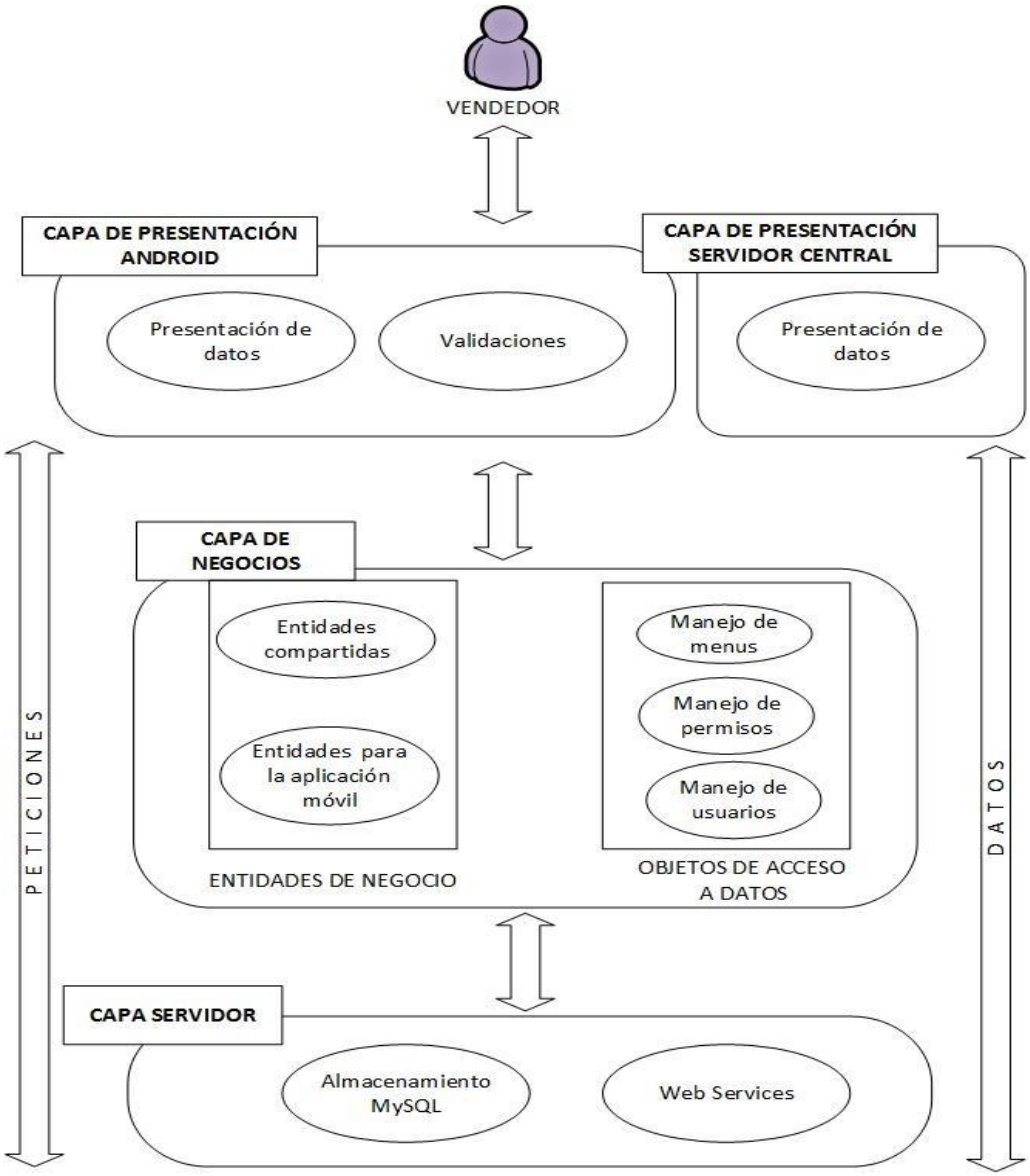


Ilustración 58. Esquema lógico de la aplicación WISE. **Fuente:** El autor.

El vendedor interactuará con la capa de presentación solicitando una petición o desencadenando acciones, dicha petición será almacenada en la base de datos móvil para luego ser atendida por la capa de negocio, la cual aplicará la lógica de tratamiento de datos. A continuación la capa servidor proporcionará un lenguaje común mediante el servicio web, de esta forma la petición será almacenada en el servidor. Por último se presentará el resultado en la interfaz de recepción de pedidos móviles, la cual se encargará del despacho de la orden. Ilustración 58.

3.3.2 Esquema Físico

El esquema físico introduce otro concepto importante denominado **niveles,** encargados de identificar cada equipo donde va a correr el código fuente y adecuarlo a una determinada capa lógica.

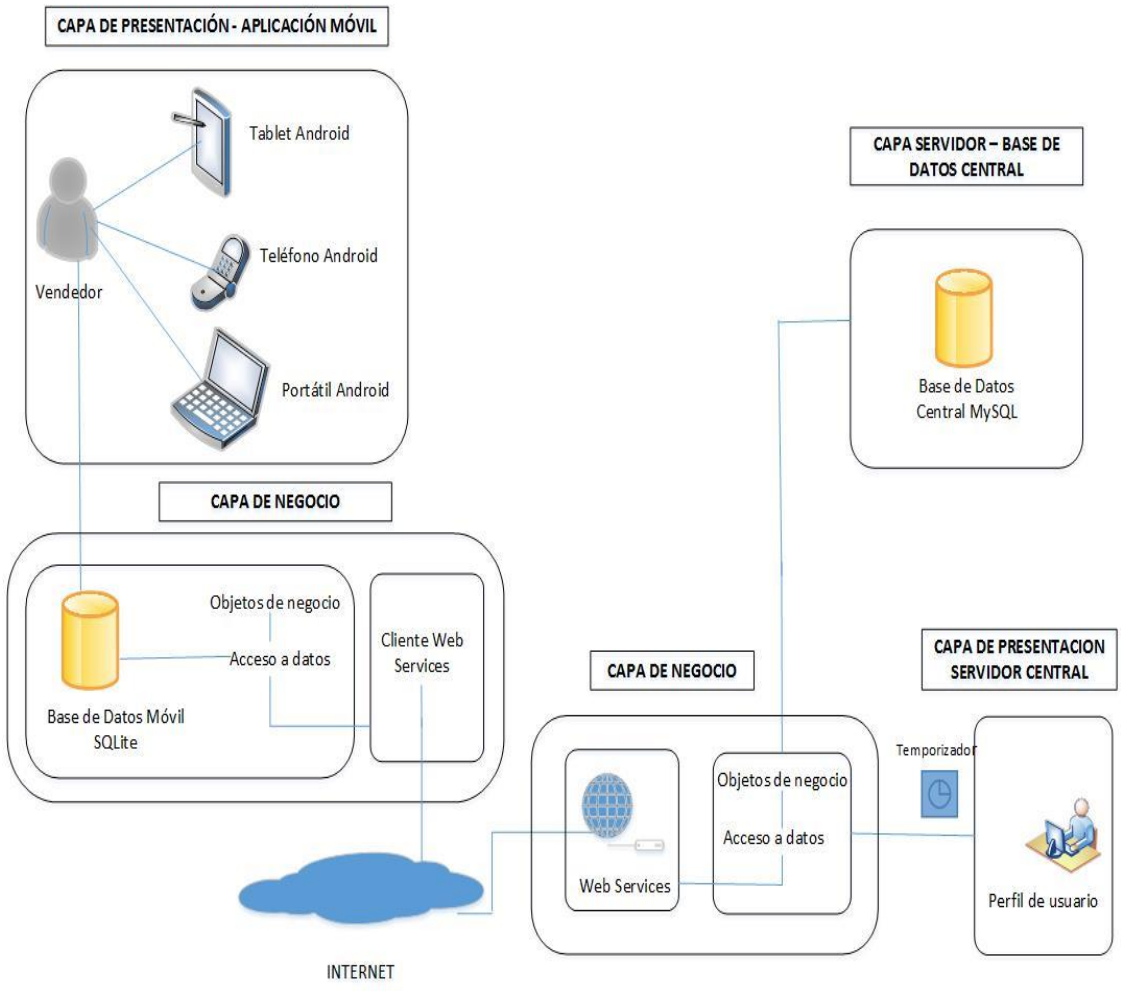


Ilustración 59. Esquema físico de la aplicación WISE. **Fuente:** El autor.

La Ilustración 59 presenta la distribución física de los elementos. En primer lugar tenemos la capa de presentación que incluye el dispositivo móvil con el cual el vendedor emitirá una orden de pedido, dicha orden será almacenada en la base de datos SQLite integrada en el dispositivo. A continuación el cliente móvil enviará su petición al servidor de aplicaciones,el cual mediante el protocolo de comunicación SOAP permitirá el intercambio de informacion con la base de datos central MySQL. Para terminar el proceso, el registro de pedido será alamacenado en la base de datos central y el perfil de usuario (contadora) se encargará de la autorización.

Considerando que un grupo lógico no contempla aspectos críticos como: almacenamiento, seguridad, sincronización, comunicación e infraestructura. El esquema físico nos ayuda a asignar estas diferentes funcionalidades en servidores separados,equipos adecuados e implementar una topología de red que se ajusta a la necesidad de la empresa.

3.3.2.1 Factores que influyen en la distribucion física

Se enumeran algunos factores que un diseñador debe tomar en cuenta a la hora de crear el plano de distribución física y asignar los elementos lógicos:

- **Infraestructura de comunicaciones:** Si la accesibilidad es un factor crítico debe asegurarse que la capa de presentación sea lo más delgada posible.Por ejemplo: implementar una arquitectura orientada a servicios (SOAP) [48].

- **Seguridad:** Implica una serie de mecanismos de protección tales como: autenticación, cifrado,encriptación de claves , perfiles de acceso,etc.

- **Concurrencia:** Una aplicación móvil es diseñada con el objetivo de ser utilizada por varias personas,esto implica considerar el número de usuarios que la aplicación puede soportar sin afectar su rendimiento.

- **Dependencia:** El patrón capas reduce la dependencia entre los diferentes niveles permitiendo que cada uno se comporte como un módulo independiente.Aquí la capa superior podrá acceder a los servicios ofrecidos por la inferior,pero el nivel más bajo desconoce la existencia de capa superior [48].

- **Reutilización:** En muchos ocasiones será necesario reutilizar las reglas de negocio. En estos casos dividir la aplicación en diferentes componentes físicos que cumplan funciones específicas es muy útil ya que aportará flexibilidad a la hora de incorporar nuevas tenologías que reutilicen las mismas reglas de negocio.

- **Rendimiento:** La colocación de reglas a nivel de servidor puede influir altamente en el rendimiento dado que los SGBD copilan y ejecutan rápidamente estos procedimientos.

- **Impacto en las modificaciones posteriores:** El desarrolador debe considerar el impacto de alguna actualización. Si no es factible quizás no sea aconsejable utilizarlo o se deberia realizar una prueba utilizando una versión anterior.

- **Dificultad de implantación:** A pesar de las facilidades que ofrecen entornos avanzados como Visual Studio, una aplicación móvil requiere una implantación más compleja. Será necesario poner en marcha servicios, sincronizaciones y quizás atravesar fronteras de seguridad como encriptacion o cifrado de claves

3.4 Diseño del plan de experimentación y pruebas

Las pruebas del software han evolucionado hacia una forma más constructiva, ya no se asume que realizar pruebas empieza cuando la fase de programación a concluido,las pruebas se ven ahora como una actividad presente durante todo el ciclo de vida del proyecto [39,41].

Si bien las pruebas no tienen como prioridad prevenir errores sino detectarlos, la prevención también juega un papel importante para disminuir defectos y es ideal para cubrir todas las posibles situaciones por las que podría atravesar la aplicación.

La relevancia de esta fase dependerá de los criterios y estrategias que se apliquen para encontrar el mayor número de errores posibles, ya que mientras más tiempo tarde el desarrollador en identificar un error más costoso será el mantenimiento del sistema.

3.4.1 Tipos de Pruebas

Conceptualmente se puede distinguir tres grandes niveles de pruebas, llamadas de Unidad, de Integración y Funcionalidad. Sin embargo, también puede hacerse pruebas a muchas propiedades no funcionales como rendimiento, seguridad, confiabilidad, facilidad de uso, entre otras [39].

La finalidad de aplicar pruebas a diferentes niveles es comprobar las especificaciones funcionales de cada módulo, verificar propiedades diferentes y alcanzar un determinado objetivo. Cabe recalcar que en todos los niveles de pruebas, el mayor desafío es identificar las entradas y salidas, esto implica un análisis riguroso de los aspectos críticos a evaluar y el uso de técnicas de recolección de datos.

Si bien cada prueba tiene un papel específico no podemos disminuir su importancia, ya que un resultado exitoso dependerá de la ejecución de mínimo dos pruebas para obtener un resultado con credibilidad. En la Tabla 29 observamos el esquema de pruebas que se aplicará para evaluar la aplicación móvil.

TIPO DE PRUEBA	DESCRIPCION	PARTICIPANTES	METODO
Usabilidad	Detectar errores de interfaces y evaluar la experiencia del usuario.	Desarrolladores Vendedores Equipo supervisión de Agrota.	Aplicación del Test de Usabilidad, Lluvia de ideas.
Funcionalidad	Detectar errores en la lógica, funcionamiento y algoritmos.	Desarrolladores	Método de Caja Blanca.

Tabla 29. Niveles de Prueba del Software. **Fuente:** El autor.

3.4.1.1 Pruebas de usabilidad

Una de las mejores forma de evaluar la usabilidad de un producto es poniéndola a prueba con usuarios reales para esto el vendedor deberá usar las diferentes interfaces y registrar una orden de pedido exitosa e indicar si la aplicación contribuye a mejorar su trabajo operativo.

El siguiente test tiene como objetivo evaluar la percepción del usuario al usar la aplicación WISE y analizar 5 puntos críticos que determinarán:

Puntos a evaluar	**Descripción**
Contenido:	Aspectos relacionados a la distribución del contenido y los formatos utilizados para mostrar la información al usuario.
Facilidad de Aprendizaje:	Evaluar la facilidad que le resulta al usuario utilizar la aplicación por primera vez.
Accesibilidad:	Consideraciones visuales, físicas o auditivas.
Satisfacción:	Indicar el grado de satisfacción del usuario.
Eficiencia:	Evaluar la adaptación de la aplicación frente al uso de usuarios externos además de registrar el tiempo que toma emitir una orden de pedido.

Tabla 30. Criterios a evaluar al aplicar el test de usabilidad. **Fuente:** El autor.

- **Participantes:** Se tomará una muestra de 7 vendedores de la empresa "Agrota Cía. Ltda". La prueba de usabilidad se llevará a cabo bajo la supervisión del equipo de Agrota y los desarrolladores.

- **Criterios de aceptación o rechazo**

  - **Errores leves:** Confusión del usuario con algunas imágenes.

  - **Errores medios:** Falta de personalización para presentar los datos numéricos (decimales), no validar campos en blanco.

  - **Errores críticos:** La sincronización no garantiza que la información del servidor sea igual a la información del móvil, se almacena información errónea en la base de datos, inconsistencia de datos, el proceso de actualización no es satisfactorio.

Nota: La aplicación WISE será aprobada si alcanza el rango entre 85% al 100% de aceptación. Caso contrario si no llega a cumplir con un promedio del 85%, el Test de Usabilidad será reprobado hasta su posterior corrección.

TEST DE USABILIDAD

Presentación: Le agradecemos su disposición al participar en esta "Prueba de Usabilidad "que nos ayudará a evaluar la aplicación móvil que estamos desarrollando y detector posibles fallas que éste tenga.

Nombres:		Edad:	
Apellidos:		Fecha de Prueba:	
Profesión:		Resultados():	

Instrucciones:

Lea detenidamente cada una de las preguntas y seleccione con una X la opción que más se ajuste a su opinión sobre la usabilidad de la aplicación móvil WISE.

Parte I: CONTENIDO

1. Le parece adecuada la selección de contenido presentado en el menú principal.

SI	
NO	

→ **¿Por qué? ----------**

---

2. Al hacer click en los módulos de la aplicación ¿Halló en la información ofrecida lo que esperaba?

SI	
NO	

→ **¿Por qué? ----------**

---

3. ¿Existen descripciones dentro de cada módulo que le permiten saber exactamente donde se encuentra y con qué cliente está trabajando?

SI	
NO	

→ **¿Por qué?** ----------

---

4. Cree que los textos introductorios, cómo títulos, información de productos son claros y de fácil lectura.

SI	
NO	

→ **¿Por qué?** ----------

Parte II: FACILIDAD DE APRENDIZAJE

5. Las imágenes presentadas en la aplicación son reconocibles y describen claramente su función.

SI ☐

NO ☐ → **¿Cuáles?** ----------

---

6. Considera que las opciones secundarias de desplazamiento como: menú, opciones de subir, atrás, le ayudaron a usar la aplicación de mejor manera.

SI ☐

NO ☐ → **¿Por qué?** ----------

---

7. Los mensajes que presenta la aplicación son fáciles de entender.

SI ☐

NO ☐ → **¿Cuáles?** ----------

Parte III: ACCESIBILIDAD

8. Considera que el proceso de actualización de la cartera del cliente es satisfactorio.

SI ☐

NO ☐ → **¿Por qué?** ----------

---

9. El proceso de sincronización le garantiza a usted que los datos del móvil son iguales a los del servidor central.

SI ☐

NO ☐ → **¿Por qué?** ----------

Parte IV: SATISFACCIÓN

10. Existe algún elemento grafico o texto que le ha causado confusión.

SI	
NO	

→ **¿Cuáles?** ----------

---

11. La aplicación maneja estilos en cada una de sus pantallas.

SI	
NO	

→ **¿Cuáles?** ----------

---

12. La información de detalle de la orden de pedido es legible y correcta.

SI	
NO	

→ **¿Por qué?** ----------

Parte V: EFICIENCIA

13. Cree usted que la aplicación móvil ha simplificado su trabajo operativo y ha agilizado el proceso de registro de pedidos.

SI ☐

NO ☐ → **¿Por qué?** ----------

---

14. En su opinión, la aplicación móvil facilita la comercialización de los productos y mejora la productividad de la empresa.

SI ☐

NO ☐ → **¿Por qué?** ----------

3.4.1.2 Pruebas de Funcionalidad

Para comprobar la funcionalidad de la aplicación WISE se ha optado por la especificación de un plan de pruebas utilizando un método de caja blanca. El objetivo es evaluar aspectos críticos de la aplicación, identificar errores hasta ahora no visibles y documentar los resultados esperados.

A continuación se detalla el diseño de los nueve casos de prueba, los mismos que fueron creados partir de los requerimientos impuestos por las partes interesadas y serán aplicados en el capítulo 5 para verificar el nivel de cumplimiento.

CASO DE PRUEBA – 001:	INGRESAR AL SISTEMA	
Autores:	Johanna Picón, Sebastián Zhinin	
Objetivo de la Prueba:	Asegurar el acceso a la aplicación WISE solo al personal autorizado.	
Actor:	Desarrolladores	
Condiciones de Éxito:	▪ Comprobar conexión con la ip del servidor central. ▪ Registrar el dispositivo móvil en el servidor central. ▪ El sistema no permitirá dejar campos en blanco. ▪ La aplicación no permitirá acceso a usuarios no registrados. ▪ Verificar el número de intentos que puede realizar el usuario para ingresar al sistema. ▪ Verificar la emisión de un mensaje al realizar la primera sincronización con el servidor.	
Puntos de Observación:	▪ Emitir un mensaje probando conectividad con el servidor central. ▪ Si el dispositivo móvil no ha sido registrado la aplicación no le permite acceso. ▪ Si el usuario deja campos en blanco, la aplicación emitirá un mensaje de error. ▪ Si el usuario o contraseña no coincide con el servidor central la aplicación no permite autenticarse. ▪ Verificar el bloqueo del usuario al ingresar tres intentos fallidos. ▪ Emitir un mensaje al realizar la primera sincronización para descargar los datos del servidor.	
Resultados Obtenidos:	**1.** La aplicación emite un mensaje indicando conexión con el servidor. **2.** Si la tablet no está previamente registrada la aplicación no permite el acceso. **3.** Los campos en blanco están validados. **4.** Al ingresar un usuario y contraseña incorrecta, la aplicación no permite el acceso. **5.** La clave ha sido bloqueada luego realizara tres intentos inválidos. **6.** La aplicación emite un mensaje de aviso al realizar la primera sincronización.	

Tabla 31. Caso de Prueba-001: Ingresar al sistema. **Fuente:** El autor.

CASO DE PRUEBA – 002:	SINCRONIZAR DATOS DEL SERVIDOR CENTRAL	
Autores:	Johanna Picón, Sebastián Zhinin	
Objetivo de la Prueba:	Realizar la primera sincronización con el servidor para descargar los datos necesarios.	
Actor :	Desarrolladores	
Condiciones de Éxito:	▪ Verificar la emisión de un mensaje informativo indicando el inicio de descarga de información. ▪ Verificar la emisión de un mensaje de sincronización exitosa.	
Puntos de Observación:	▪ La aplicación emite un mensaje de sincronización exitosa. ▪ Los registros de eventos del servidor no presentan errores.	
Resultados Obtenidos:	**1.** La aplicación emite un mensaje indicando que la sincronización se estableció con éxito. **2.** El archivo de eventos del servidor no presenta errores.	

.

Tabla 32. Caso de Prueba-002: Ingresar al sistema **Fuente:** El autor

CASO DE PRUEBA – 003:	LISTAR CLIENTE	
Autores:	Johanna Picón, Sebastián Zhinin	
Objetivo de la Prueba:	Listar la información de detalle del cliente	
Actor:	Desarrolladores	
Condiciones de Éxito:	▪ La aplicación lista todos los clientes registrados en el servidor central.	
Puntos de Observación:	▪ La información del cliente debe mostrar el siguiente detalle: nombres, apellidos, nombreComercial, teléfono, email.	
Resultados Obtenidos:	**1.** La aplicación emite una lista con toda la información del cliente.	

Tabla 33. Caso de Prueba-003: Listar cliente. **Fuente:** El autor.

CASO DE PRUEBA – 004:	REGISTRAR ORDEN DE PEDIDO	
Autores:	Johanna Picón, Sebastián Zhinin	
Objetivo de la Prueba:	Permitir al vendedor registrar una orden de pedido exitosamente.	
Actor :	Desarrolladores	
Condiciones de Éxito:	▪ Verificar que el vendedor seleccione un cliente antes de intentar registrar la orden de pedido. ▪ Verificar que la orden contenga almenos un producto o item. ▪ El vendedor solo podrá seleccionar un item de cada grupo del combo. ▪ El vendedor está obligado a ingresar la observación de la orden de pedido. ▪ No se podrá dejar cero en el valor de la cantidad de ítems al momento de registrar una orden. ▪ Emitir un mensaje al grabar la orden de pedido.	
Puntos de Observación:	▪ Emitir un mensaje al no seleccionar previamente un cliente para registrar la orden. ▪ Emitir un mensaje indicando que la orden no contiene ítems. ▪ Emitir un mensaje de error al intentar seleccionar 2 o más ítems del grupo del combo. ▪ Emitir un mensaje al dejar en blanco el campo de observación de la orden. ▪ Emitir un mensaje cuando la cantidad de ítems de la orden sea cero. ▪ Emitir un mensaje de pedido guardado exitosamente.	
Resultados Obtenidos:	**1.** La aplicación emite un mensaje al no seleccionar un cliente antes de registrar la orden. **2.** Si la orden de pedido no contiene ítems se emite un mensaje de error. **3.** Al intentar escoger dos ítems de un mismo grupo del combo la aplicación emite un mensaje de error. **4.** Se emite un mensaje de error cuando no se ha ingresado la observación de la orden. **5.** Se emite un mensaje al dejar en cero el valor de la cantidad de ítems de la orden de pedido. **6.** La aplicación emite un mensaje de pedido grabado exitosamente.	

Tabla 34. Caso de prueba-004: Registrar Orden de Pedido. **Fuente:** El autor.

CASO DE PRUEBA – 005:	LISTAR PRODUCTOS	
Autores:	Johanna Picón, Sebastián Zhinin	
Objetivo de la Prueba:	Permitir listar productos de acuerdo a una categorización.	
Actor:	Desarrolladores	
Condiciones de Éxito:	▪ Listar productos por categorías, líneas o marcas. ▪ Identificar en qué categorización está el producto. ▪ Verificar la descripción de detalle del producto.	
Puntos de Observación:	▪ Se emite un mensaje indicando la categorización a la que pertenece el producto. ▪ El detalle del producto presenta la siguiente información: nombre, descuento, PVP, precioCliente, codigoProducto.	
Resultados Obtenidos:	**1.** La aplicación emite una lista con los productos existentes en esa categorización. **2.** La aplicación muestra descripciones indicando la categorización a la que pertenece el producto. **3.** Cada producto presenta el siguiente detalle: nombre, descuento, PVP, precioCliente, codigoProducto.	☑ ☒ ☑ ☒ ☑ ☒

Tabla 35. Caso de prueba-005: Listar Productos. **Fuente:** El autor.

CASO DE PRUEBA – 006:	AGREGAR PRODUCTOS POR CATALOGO A LA ORDEN DE PEDIDO	
Autores:	Johanna Picón, Sebastián Zhinin	
Objetivo de la Prueba:	Permitir agregar productos por catálogo directamente al pedido.	
Actor:	Desarrolladores	
Condiciones de Éxito:	▪ Comprobar la emisión de un mensaje al agregar productos correctamente al pedido.	
Puntos de Observación:	▪ Se emite un mensaje cuando la orden de pedido no contiene productos. ▪ Verificar el descuento por líneas de cada producto. ▪ Verificar que el producto tenga PVP para poder agregarlo a la orden. ▪ Verificar si el producto tiene extras o combo.	
Resultados Obtenidos:	**1.** La aplicación emite un mensaje de confirmación al momento de agregar items a la orden. **2.** Si la orden de pedido no contiene almenos un item, se emite un mensaje de error. **3.** El descuento de cada producto se aplica por línea. **4.** Todo producto tiene un precio de venta al público (PVP). **5.** Si el producto tiene extra, la aplicación muestra el cómo asociado.	☑ ☒ ☑ ☒ ☑ ☒ ☑ ☒ ☑ ☒

Tabla 36. Caso de prueba-006: Agregar productos pro catalogo a la orden de pedido.

CASO DE PRUEBA – 007:	LISTAR ORDEN DE PEDIDO	
Autores:	Johanna Picón, Sebastián Zhinin	
Objetivo de la Prueba:	Comprobar que el sistema liste las ordenes de pedido y permita visualizar el estado de la orden.	
Actor:	Desarrolladores	
Condiciones de Éxito:	▪ El sistema emite una lista con todos los pedidos registrados, permitiendo aplicar un filtro por fecha o por cliente. ▪ El sistema permita visualizar el cambio de estado del pedido (pendiente o enviado).	
Puntos de Observación:	▪ Se emite un mensaje indicando que no existen pedidos en ese rango de fecha. ▪ Se emite un mensaje al no tener órdenes asociadas a ese cliente.	
Resultados Obtenidos:	**1.** La aplicación lista todos los pedidos filtrados por fecha o por cliente. **2.** Si no existen pedidos en un rango de fecha se emite un mensaje informativo. **3.** Si el cliente no tiene pedidos registrados se emite un mensaje indicándolo. **4.** La aplicación permite visualizar el cambio de estado del pedido de pendiente a enviado.	

Tabla 37. Caso de prueba-007: Listar orden de pedido **Fuente:** El autor.

CASO DE PRUEBA – 008:	LISTAR CARTERA DEL CLIENTE	
Autores:	Johanna Picón, Sebastián Zhinin	
Objetivo de la Prueba:	Listar la cartera del cliente por un rango de fecha.	
Actor:	Desarrolladores	
Condiciones de Éxito:	▪ El sistema permite listar la cartera de un cliente activo.	
Puntos de Observación:	▪ Se emite un mensaje indicando que el cliente no tiene una cartera activa.	
Resultados Obtenidos:	**1.** La aplicación lista la cartera del cliente activo **2.** Si un cliente no tiene cartera, la aplicación emite un mensaje indicando.	

Tabla 38. Caso de prueba-008: Listar orden de pedido. **Fuente:** El autor.

CASO DE PRUEBA – 009:	ACTUALIZAR CARTERA DEL CLIENTE	
Autores:	Johanna Picón, Sebastián Zhinin	
Objetivo de la Prueba:	Permitir actualizar el detalle de cartera de un cliente concreto.	
Actor:	Desarrolladores	
Condiciones de Éxito:	▪ El sistema permite actualizar la cartera de un cliente específico.	
Resultados Obtenidos:	▪ Se emite un mensaje indicando la actualización exitosa de la cartera del cliente.	
Resultados Obtenidos:	**1.** La aplicación móvil permite actualizar la cartera de un cliente específico y no todo el estado.	

Tabla 39. Caso de prueba-009: Actualizar cartera del cliente. **Fuente:** El autor.

3.5 Parámetros de diseño para aplicaciones móviles

Otro aspecto clave en el diseño es la usabilidad que una aplicación pueda tener, esto significa que la interfaz de usuario debe ser amigable y no causar confusión. Tomando en cuenta la presencia de Android en una variedad de dispositivos y tamaños, esto implica que el desarrollador haga uso de las mejores prácticas de diseño, patrones y estándares que aporten flexibilidad a la aplicación móvil.

A continuación se expone ciertos parámetros que facilitarán el trabajo de diseño:

- **Dispositivo específico:** Una buena práctica de diseño consiste en agrupar los dispositivos de acuerdo a las características de interacción (táctil, no táctil, funcionalidad, resolución). De esta forma el desarrollador podrá realizar las pruebas pertinentes a cada grupo e identificar la solución que más se ajusta a su necesidad.

- **Usar patrones de interacción:** Un patrón describe una solución a un problema que ocurre infinidad de veces, de tal modo que pueden ser utilizado como una guía, no como una ley [51].

Android incluye una variedad de patrones de interacción que pueden agilizar el diseño de una interfaz, estos son detallados a continuación:

- ✓ **Action Bar:** Su funcionamiento es muy similar a la de un banner de los sitios web, con el logo o título generalmente a la izquierda y los elementos de navegación a la derecha. [46]. En la Ilustración 18 podemos identificar las partes del patrón Action Bar.

    1. **Icono de la aplicación:** Identifica el propósito de la aplicación.
    2. **View Control:** Diseñado para insertar vistas que faciliten el acceso al contenido.
    3. **Action Button:** Agiliza la búsqueda de contenido.
    4. **Despliegue de acciones:** Facilita la navegación o el filtrado de información.

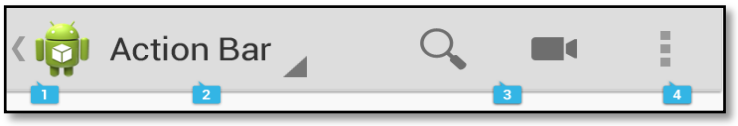


Ilustración 60.Patrón de diseño Action Bar en Android [52].

- ✓ **Navegabilidad** En Android las pestañas no deben ser elementos persistentes en la interfaz y siempre debe realizarse la navegación a una pantalla nueva [53].

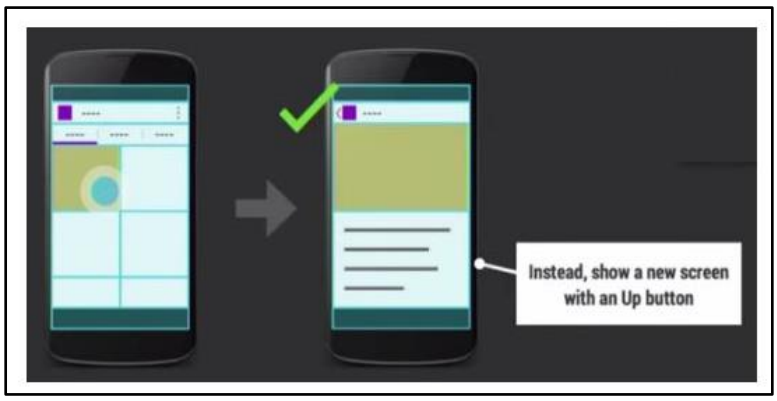


Ilustración 61. Manejo de pestañas y navegabilidad en Android [53].

- ✓ **Galería de Imágenes:** Resuelve el problema de tener que navegar entre diferentes capas. Si bien la disposición de las imágenes está delimitada por la grilla, puede dividirse en submúltiplos para ordenar mejor las imágenes y aportar simplicidad a la aplicación.

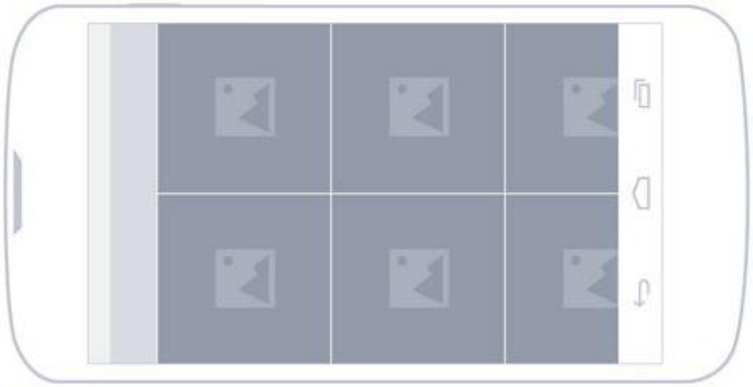


Ilustración 62. Galería o patrón de diseño dashboard en Android [37].

- ✓ **Listas**: Permiten mostrar tanto textos como imágenes, pero es importante siempre jerarquizar su contenido. Este patrón complementa la navegación a medida que el usuario se va desplazando por el contenido de la aplicación móvil.

Ilustración 63. Uso de listas en Android [37].

- **Resolución de la pantalla:** Android divide la pantalla en cuatro densidades básicas: LDPI (bajo), MDPI (medianas), IPAP (alto), y XHDPI (extra alta). Esto es importante cuando se requiere entregar todos los elementos gráficos (mapas de bits) en grupos de diferentes densidades y usar varios dispositivos [37].

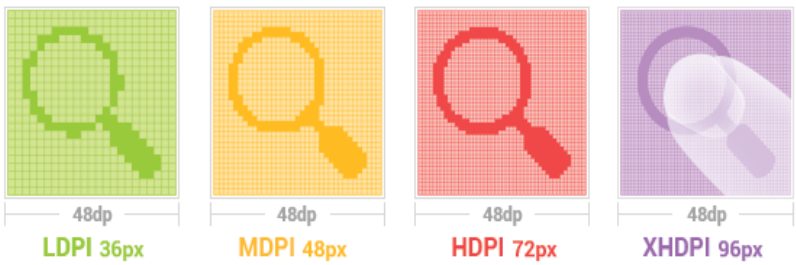


Ilustración 64. Icono de buscar diseñado para diferentes densidades [37].

- **Utilizar elementos visibles:**

  - ✓ **Botones de teclado**: Ejemplo: Controlar la tecla back que se utiliza para navegar hacia atrás cronológicamente por el histórico de las pantallas que ha utilizado el usuario.

  - ✓ **Patrones de interacción comunes:** La implementación de un botón que permita realizar acciones similares como: guardar, agregar, actualizar.

  - ✓ **Colores:** Existen ciertos colores reservados como el rojo para advertencias o errores, el verde para mensajes de éxito o confirmación y el amarrillo para prevención [37].

- **Tipografía**

La pantalla influye mucho en el comportamiento y desempeño tipográfico si se tiene en cuenta que, en algunos casos es sumamente pequeña como es el caso de un celular o más amplia como en una tableta.

En Android, el tamaño tipográfico se mide en sp (píxeles escalados), permitiendo ajustar o corregir la fuente de acuerdo al tamaño de pantalla. Los tamaños más comunes van desde 12sp hasta 22sp [37].

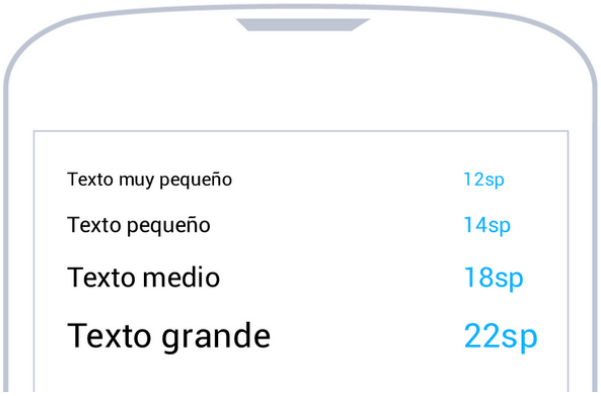


Ilustración 65. Diferentes tamaños de fuentes en Android de acuerdo al uso y jerarquía de los elementos [37].

CAPÍTULO 4

4. IMPLEMENTACIÓN

Introducción

La fase de implementación es considerada la etapa más larga del ciclo de vida del software,aquí el programador se encarga de dar vida a los diseños en papel y crea la estructura base sobre la cual se apoyará el funcionamiento de la aplicación.

El desarrolador define las convenciones de programación (métodos, variables, clases, paquetes) necesarios para crear una version inicial. Luego dedica gran parte de su tiempo a corregir los errores funcionales para preparar a la aplicación para su aprobación.

Es por ello que la fase de construcción se encuentra intimamente ligada con el diseño y la verificación, es en este punto donde la ejecución pruebas ayudan a asegurar la calidad y desempeño de la app.

Propósito

Permitir al vendedor interactuar con las diferentes pantallas de una forma física, probar la funcionalidad y sobre todo evaluar la usabilidad del sistema final.

Es importante considerar que la aplicación pueda responder a las necesidades del usuario, para esto se implementará una guía de fácil acceso compuesta por iconos, textos descriptivos, títulos, fondos de pantalla, colores consecuentes, elementos del menú, etc.

También se considerará la integración de la aplicación móvil con el sistema central y las herramientas de apoyo para crear aplicaciones sin conexión a internet. Dado que los usuarios dan mayor valor a aquellas herramientas que permiten simplificar su vida ,desarrollar una aplicación offline representa para el vendedor portabilidad y conectividad desde cualquier sitio.

4.1 Implementación de la interfaz de usuario

El diseño de la interfaz de usuario es la categoría de diseño que establece un medio de comunicación entre el hombre y la máquina [43].Theo Mantel establece tres reglas de oro que forman la base del diseño de la interfaz de usuario:

- **Dar el control al usuario:** Crear una interacción flexible es decir no obligar al usuario a realizar acciones innecesarias y no deseadas [43].

- **Reducir la carga de memoria del usuario:** La interfaz no debe poner a prueba la memoria del usuario sino agregar claves visules para recordar acciones anteriores.

- **Construir una interfaz consecuente:** La informacion visual debe estar organizada deacuerdo a un estándar gráfico manejado en todas las pantallas.

4.1.1 Interfaz de ingreso al sistema – Autentificación

La pantalla de ingreso a la aplicación WISE contiene el formulario de logueo basado en un usuario y contraseña. Al presionar el botón *Ingresar* la aplicación realiza el proceso de autenticación con el servidor central. Si el usuario se encuentra registrado, permitirá el ingreso al menú principal caso contrario aparecerá una notificación indicando el inconveniente.

Ilustración 66. Interfaz de ingreso al sistema. **Fuente:** El autor.

1. **Ingresar usuario:** El vendedor debe ingresar su nombre de usuario previamente registrado en el servidor central.

2. **Ingresar contraseña:** El vendedor debe ingresar su password para iniciar el proceso de autenticación.

4.1.2 Interfaz del Menú Principal

Permite organizar las funciones de la aplicación de forma que puedan ser encontrados rápidamente por el usuario. El menú principal ha sido dividido en seis módulos como podemos observar en la Ilustración 67.

- Módulo Productos
- Módulo Pedido
- Módulo Clientes
- Módulo Hoja de Vista
- Modulo Reportes
- Módulo Cartera

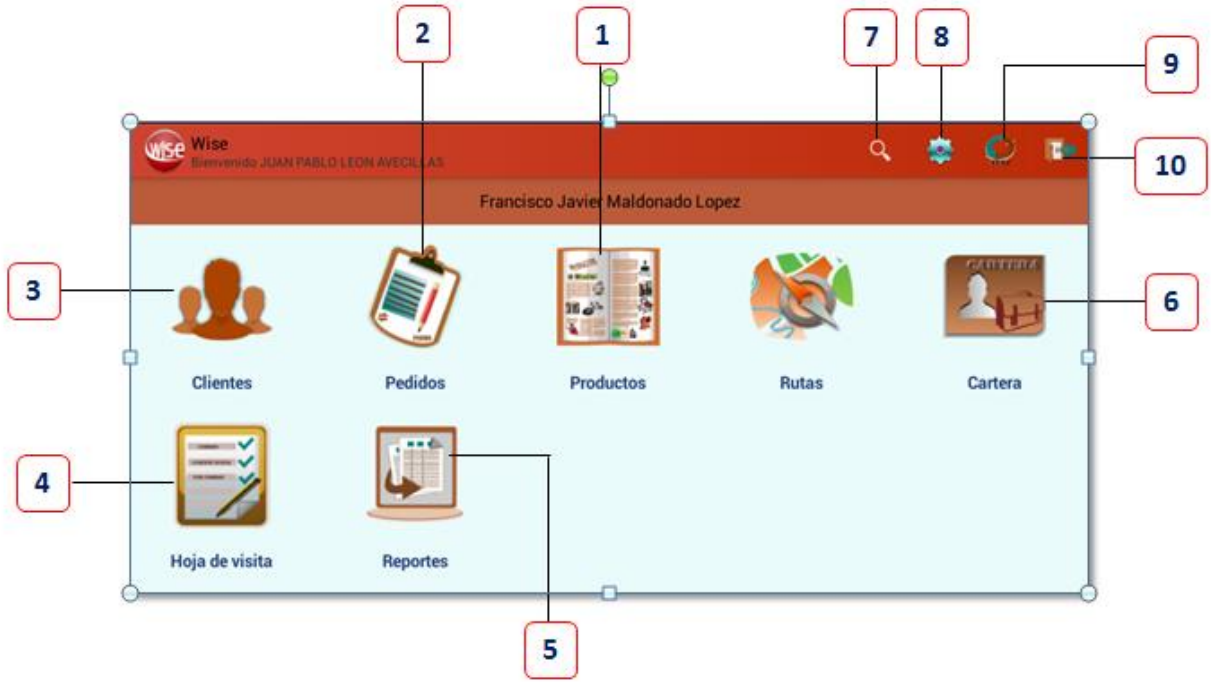


Ilustración 67. Interfaz Menú Principal. **Fuente:** El autor.

1. **Módulo Productos:** Contiene la información de los productos categorizados por líneas, categorías y marcas.

2. **Módulo Pedidos:** Permite ingresar al formulario de *Nueva Orden* y *Ver Ordenes.*

3. **Módulo Clientes:** Contiene la información del cliente seleccionado.

4. **Módulo Hoja de Visita:** Registra la información de la acción realizada por el vendedor al momento de visitar a su cliente.

5. **Módulo Reportes:**

6. **Módulo Cartera:** Muestra las facturas, créditos y los ítems vendidos al cliente seleccionado.

7. **Buscador:** Permite buscar un determinado cliente

8. **Información de la dirección IP del servidor:** Permite ver la información de configuración del servidor central (IP y puerto).

9. **Opción de sincronización:** Permite sincronizar la información nueva o los cambios realizados en el servidor central. Actualizando solo los cambios en el estado del modelo y no todos el esquema de base de datos móvil.

10. **Salir**

4.1.2.1 Interfaz Módulo Productos

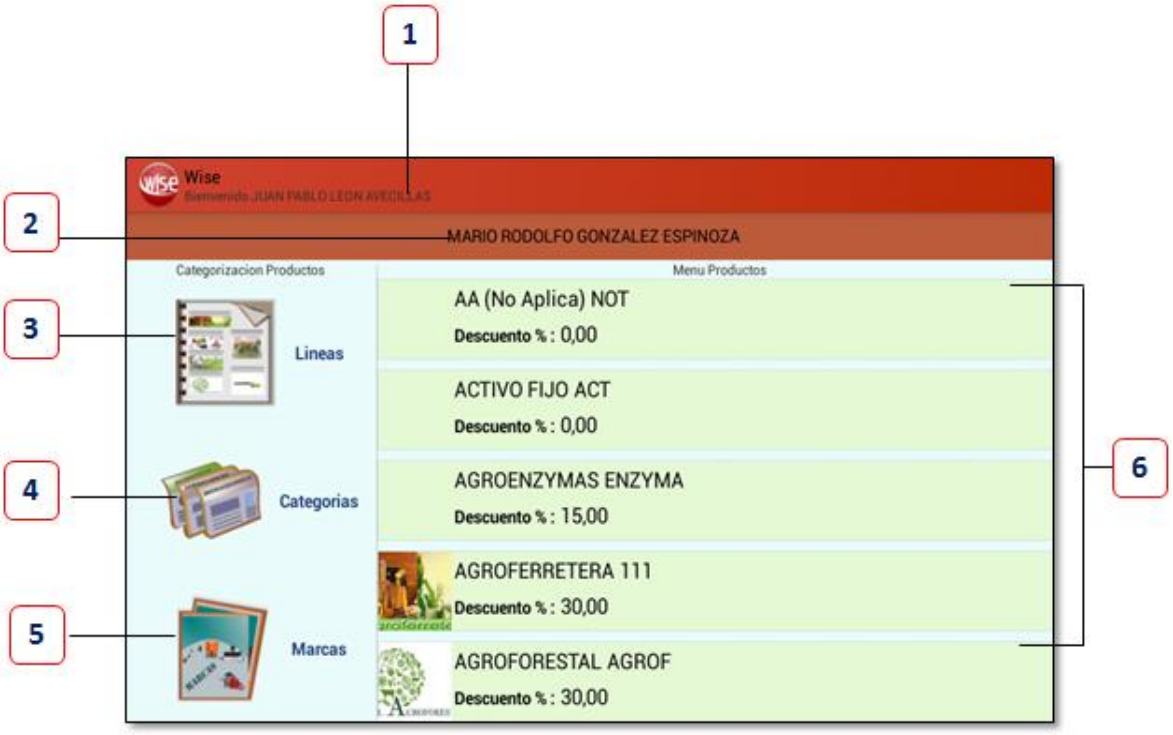


Ilustración 68. Interfaz de ingreso al sistema. **Fuente:** El autor.

1. **Nombre del vendedor.**

2. **Nombre del Cliente.**

3. **Categorización de productos por líneas:** Filtra productos de acuerdo a una línea específica. Ejemplo: agroforestal, agropecuaria, nutricional, etc.

4. **Categorización de productos categorías:** Filtra productos de acuerdo a una categoría específica. Ejemplo: desmalezadora, maquinaria, agroquímicos, etc.

5. **Categorización de productos por marcas:** Filtra productos de acuerdo a una marca específica. Ejemplo: shindaiwa, subaro, hecho, etc.

6. **Lista según la categorización de producto (líneas, categoría, marca):** Según la categorización de los productos se listará los productos incluidos. El producto elegido presentará la siguiente interfaz.

Ilustración 69. Interfaz del producto. **Fuente:** El autor.

1. **Opción de búsqueda:** Permite buscar un determinado producto.
2. **Nombre del producto:** Información del producto.
3. **PVP:** Precio de venta del producto**.**
4. **Descuento del producto:** Representa el descuento del producto asignado de acuerdo a la línea que pertenece y la calificación del cliente.
5. **Precio Cliente:** Precio del cliente una vez aplicado el descuento**.**
6. **Botón Agregar Pedido:** Permite agregar el producto a la orden de pedido.

4.1.2.2 Interfaz Módulo Pedido

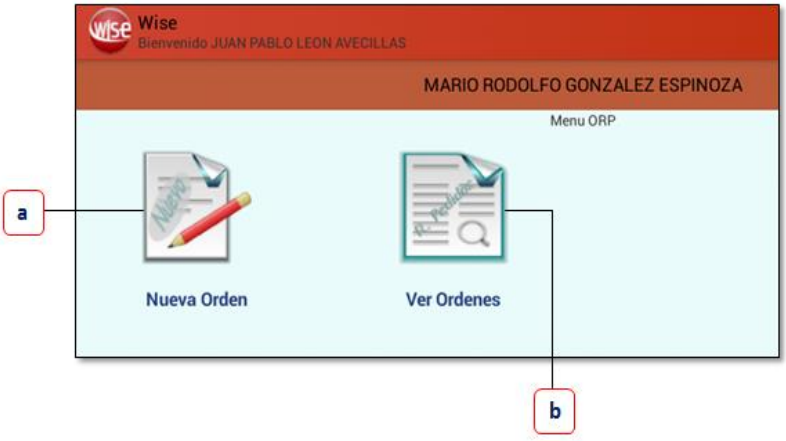


Ilustración 70. Interfaz Módulo Pedido. **Fuente:** El autor.

Al ingresar al módulo pedido se presentan dos opciones:

a) ***Nueva Orden:*** Permite registrar una nueva orden de pedido.

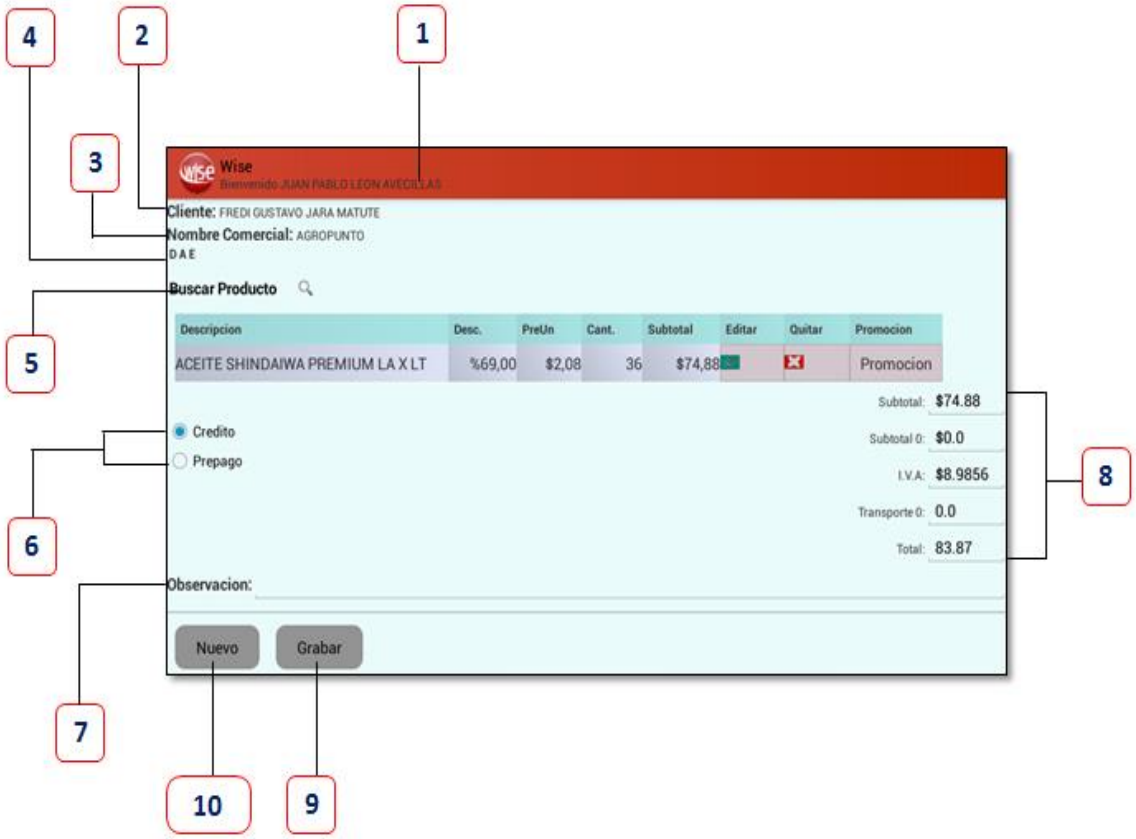


Ilustración 71. Interfaz orden de pedido. **Fuente:** El autor.

1. **Nombre del vendedor.**
2. **Nombre del cliente.**
3. **Nombre comercial del cliente.**
4. **Calificación del cliente:** La primera letra representa el volumen seguido del pago y tipo de cliente.
5. **Buscar producto:** Permite buscar productos para agregarlos a la orden.
6. **Seleccionar forma de pago.**
7. **Observación de la orden:** El vendedor debe ingresar la observación de la orden obligatoriamente para poder grabar la orden.
8. **Valor total y subtotal de la orden de pedido.**
9. **Botón Guardar:** Permite guardar el pedido.
10. **Botón Nuevo:** Permite crear una nueva orden.

b) ***Ver Órdenes:*** Permite al vendedor visualizar todas la ordenes emitidas en una fecha determinada.

Ilustración 72. Interfaz Ver Ordenes. **Fuente:** El autor.

1. **Nombre del cliente.**
2. **Fecha de búsqueda.**
3. **Botón Generar:** Permite filtrar por un rango de fecha las ordenes emitidas por el vendedor.
4. **Nombre del cliente.**
5. **Fecha de emisión de la factura.**
6. **Valor:** Monto de la compra.
7. **Estado:** Permite verificar el estado de la orden, si se encuentra en color rojo la orden aún no ha llegado a la central caso contrario cuando está en verde la orden está grabada en la central.
8. **Ver Items:** Permite ver los ítems que incluye la orden de pedido.
9. **Orden:** Incluye la información sobre en qué nivel de autorización se encuentra la orden.

4.1.2.3 Interfaz Módulo Cliente

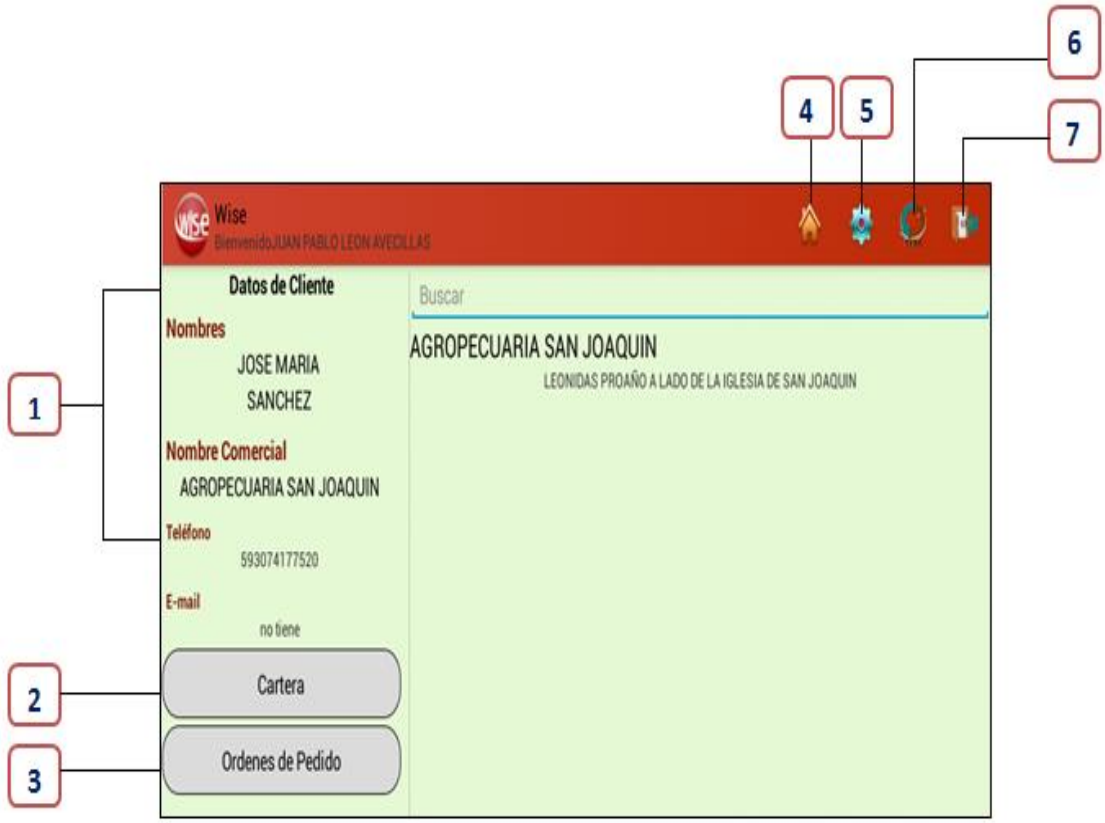


Ilustración 73. Interfaz Módulo Cliente. **Fuente:** El autor.

1. **Datos cliente:** Incluye la información del cliente como: nombres, nombre comercial, teléfono, email.
2. **Cartera:** Redirecciona al módulo cartera, aquí se podrá la información sobre las cuotas canceladas por el cliente.
3. **Ordenes de Pedido:**
4. **Opción para regresar al menú principal.**
5. **Información de la dirección IP del servidor:** Al ingresar a esta opción se encontrara la información de conexión con el servidor central como la IP y el puerto.
6. **Opción de sincronización:** Permite sincronizar los datos del servidor con los de la tablet, manteniendo al día la información del negocio.
7. **Salir.**

4.1.2.4 Interfaz Módulo Cartera

Facturas	Fecha	Valor	Retencion	Saldo	Ver Credito	Ver Items
001-002-686	2012-11-29	$ 85,80	$ 0,00	$ 0,00	Credito Detalle	Ver Items
001-001-22007	2013-03-26	$ 1746,00	$ 0,00	$ 0,00	Credito Detalle	Ver Items
001-001-22127	2013-04-10	$ 1094,89	$ 0,00	$ 0,00	Credito Detalle	Ver Items
001-001-22727	2013-06-10	$ 554,40	$ 0,00	$ 0,00	Credito Detalle	Ver Items
001-001-22807	2013-06-14	$ 39,38	$ 0,00	$ -0,00	Credito Detalle	Ver Items
001-001-22940	2013-06-26	$ 55,37	$ 0,00	$ 0,00	Credito Detalle	Ver Items
001-002-2289	2013-06-28	$ 79,01	$ 0,00	$ 0,00	Credito Detalle	Ver Items
001-001-23003	2013-07-01	$ 125,39	$ 0,00	$ 0,00	Credito Detalle	Ver Items
001-001-23172	2013-07-19	$ 117,78	$ 0,00	$ 0,00	Credito Detalle	Ver Items

Ilustración 74. Interfaz Módulo Cartera. **Fuente:** El autor.

8. **Fecha de búsqueda.**
9. **Botón Cargar Datos:** Lista la información de la cartera de un determinado cliente de acuerdo a un filtro de fecha.
10. **Número de Factura.**
11. **Fecha de facturación.**
12. **Valor de la compra.**
13. **Valor de retención.**
14. **Saldo.**
15. **Ver Crédito:** Permite visualizar el número de cuotas del cliente**,** la forma de pago además de consultar el saldo disponible.
16. **Ver Ítems:** Incluye los ítems de la factura.

4.1.2.5 Interfaz Módulo Hoja de Visita

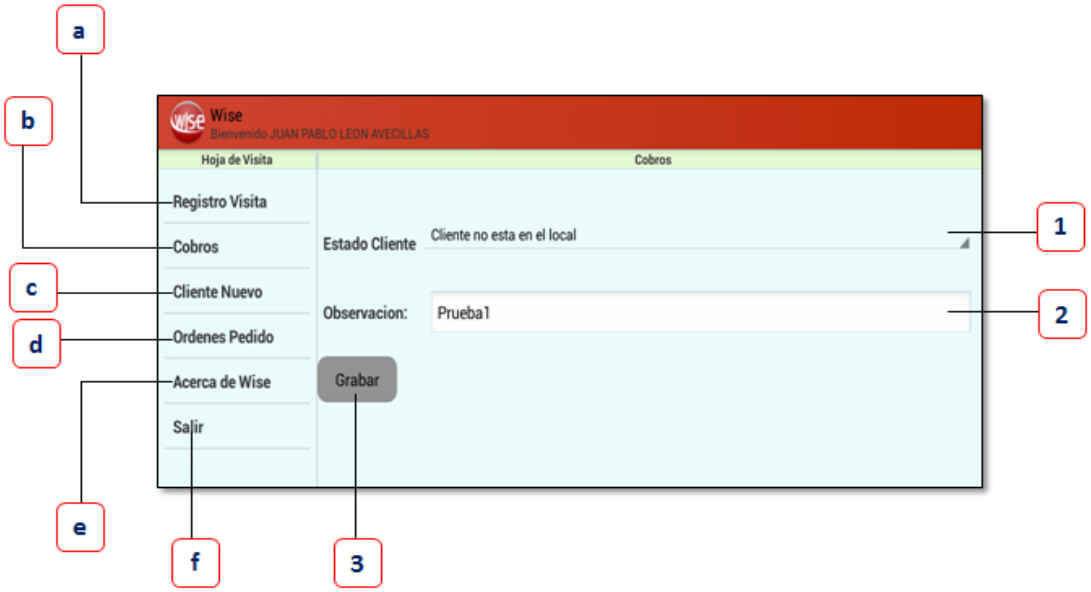


Ilustración 75.Interfaz Módulo Hoja de Visita. **Fuente:** El autor.

a) Registro de Visita: Su interfaz incluye lo siguiente:

1. **Estado del cliente:** Incluye la información del cliente al realizar la visita.
2. **Observación del registro de visita.**
3. **Botón Grabar:** Graba la información de la hoja de visita.

b) Cobros:

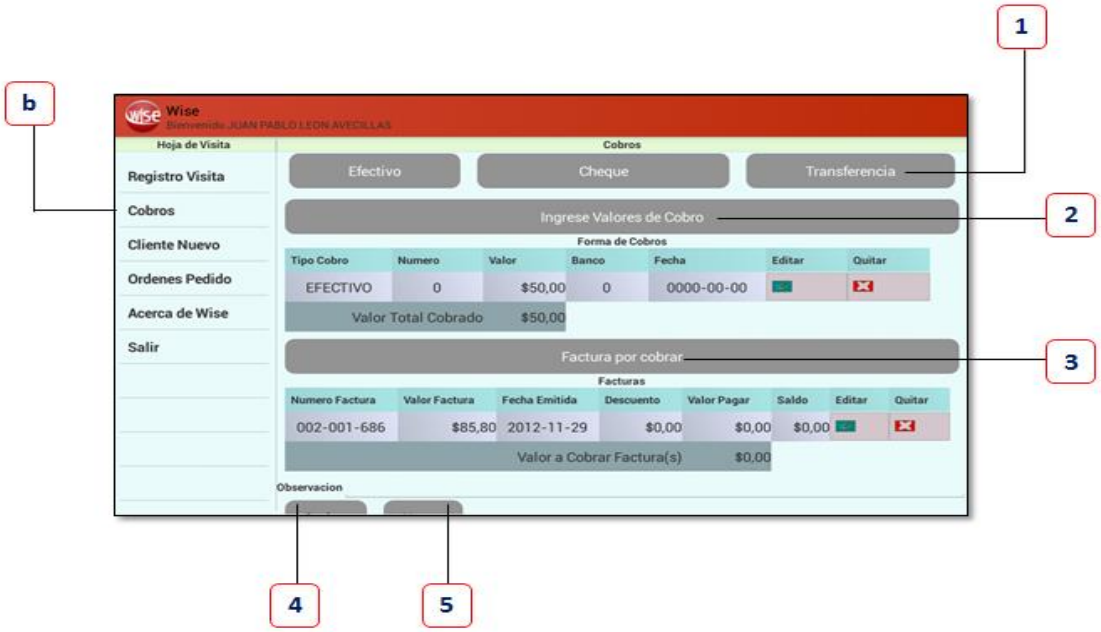


Ilustración 76. Interfaz Módulo Hoja de Visita (Cobros). **Fuente:** El autor.

1. **Forma de pago:** Incluye tres opciones de forma de pago: efectivo, cheque y transferencia. De las cuales solo se escogerá una y las demás serán bloqueadas.
2. **Valores de cobro:** Permite registrar el valor cobrado por el vendedor.
3. **Facturas por cobrar:** Muestra las facturas pendientes.
4. **Botón Guardar.**
5. **Botón Nuevo.**

c) **Cliente Nuevo:** Permite crear un cliente nuevo.
d) **Ordenes de Pedido:** Redirecciona al módulo pedido.
e) **Acerca de WISE:** Información de ayuda sobre la aplicación.
f) **Salir.**

4.1.2.6 Interfaz Modulo Reportes

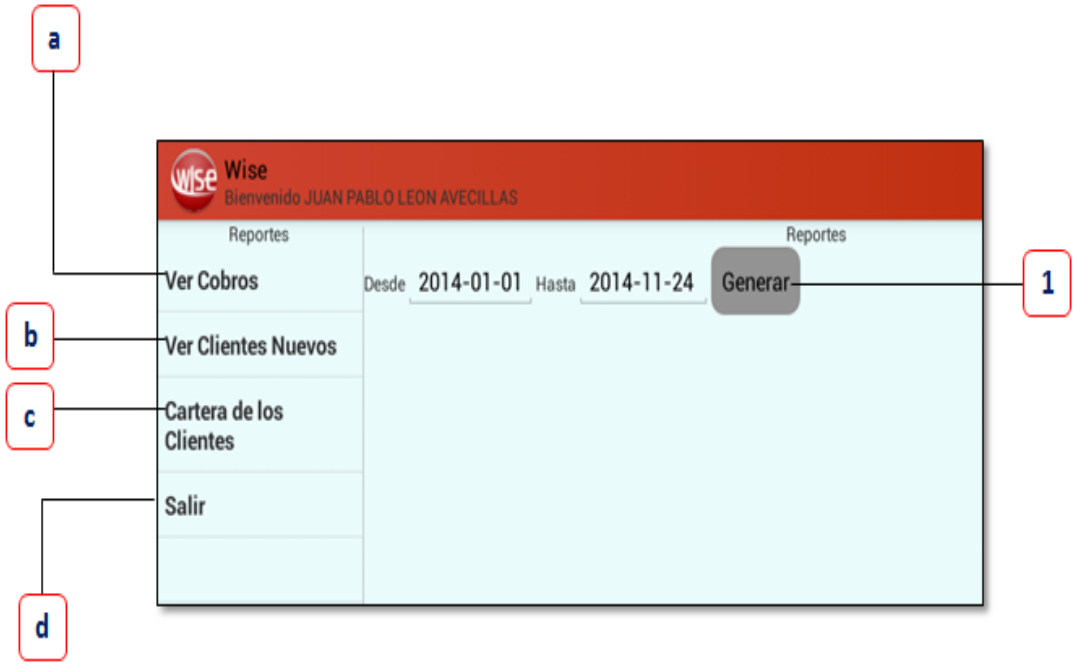


Ilustración 77. Interfaz Módulo Cartera. **Fuente:** El autor.

a) **Ver Cobros:**
   1. **Botón Generar:** Lista todos los cobros efectuados de acuerdo a un rango de fecha.
b) **Ver Clientes Nuevos:** Visualiza todos los nuevos clientes creados.
c) **Cartera de los Clientes:** Redirecciona al módulo cartera.
d) **Salir**

4.2 Creación de la Base de Datos móvil en el sistema central

SQLite es el gestor de base de datos por efecto de Android, su gran aceptación en el ámbito empresarial se debe a la simplicidad de configuración, autonomía y código libre. Su diseño va más allá del almacenamiento, permitiendo crear de aplicaciones ligeras, personales y que soportan una alta transaccionalidad [54].

Incorpora una serie de herramientas para manipulación de datos y ejecución de consultas, entre ellas la clase SQLite Helper responsable de abrir o crear la base de datos móvil así como actualización hacia nuevas versiones. Desde el punto de vista del desarrollador podemos apreciar características atractivas frente a las base de datos tradicionales como:

- Autonomía ya que utiliza un conjunto de librerías para trabajar de forma directa con la aplicación y los datos.

- Altamente transaccional en el cual todos los cambios y consultas se realizan de forma automática y consistente.

- Auto contenida, esto significa que requiere un mínimo de soporte para librerías externas, ideal para aplicaciones que deben correr sin modificaciones en una variedad de dispositivos [55].

- Almacenamiento persistente de objetos, configuraciones y preferencias de usuario.

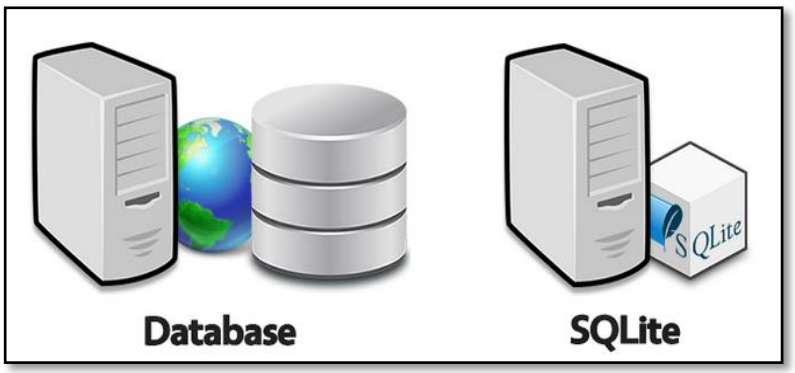


Ilustración 78.SQLite frente a otras base de datos [55].

4.2.1 Esquema Lógico de Base de Datos

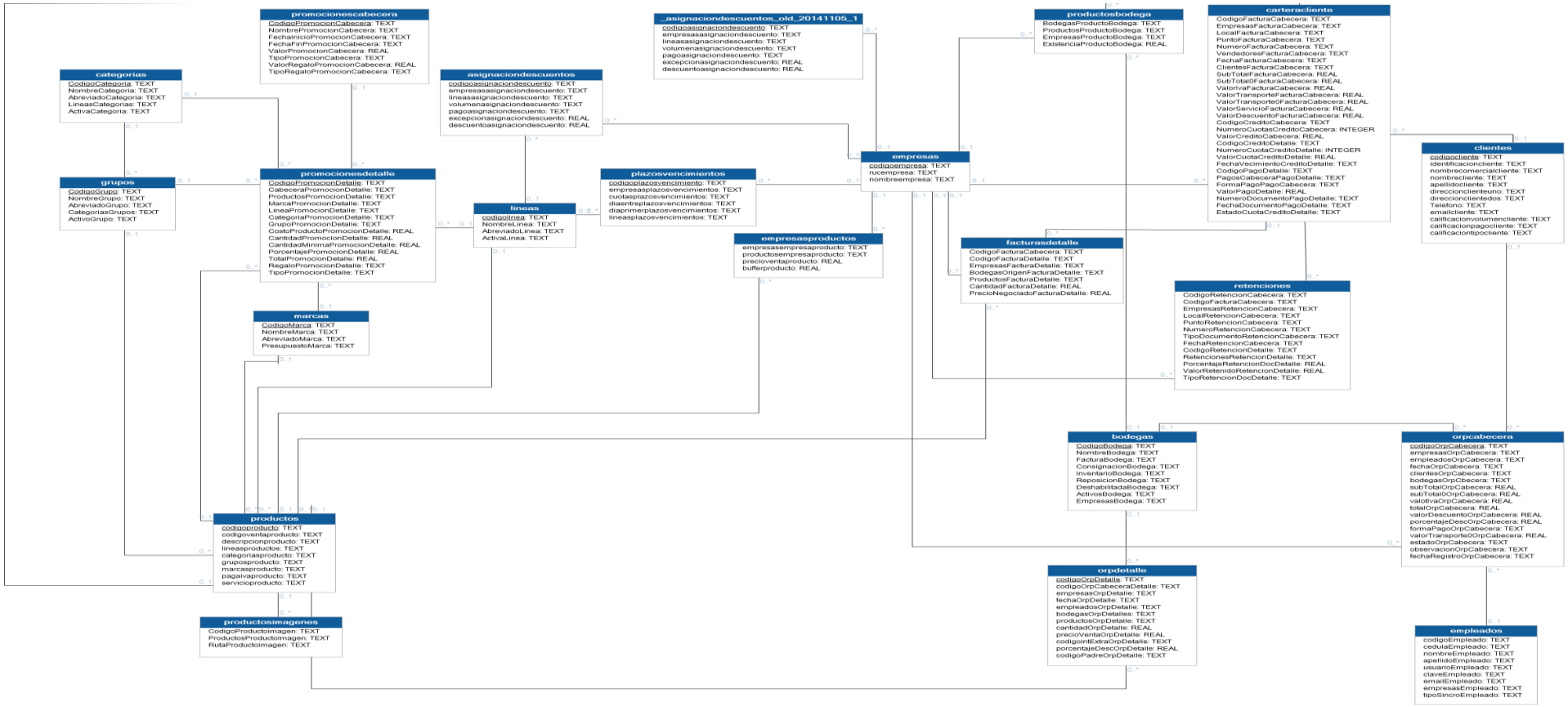


Ilustración 79. Diagrama Entidad Relación del móvil Android. **Fuente:** El autor

4.3 Integración de la aplicación móvil con el sistema central

La integración de la aplicación móvil con el sistema central lo define el Web Services, su función es compartir todos los datos, procesos y lógica de negocios alojados en el servidor de aplicaciones.

Considerando el ambiente corporativo en el que nos desenvolvemos el intercambio de información requiere un nivel de seguridad extra para no comprometer los datos críticos de la empresa. Es por esta razón que el intercambio de mensajes se realizará mediante el protocolo SOAP.

Si bien SOAP posee una arquitectura bastante compleja, los beneficios son grandes ya que incentiva la reusabilidad, la estandarización basada en XML, la independencia de plataformas, ideal para aplicaciones que implementan una arquitectura distribuida en capas [56]. Ilustración 80.

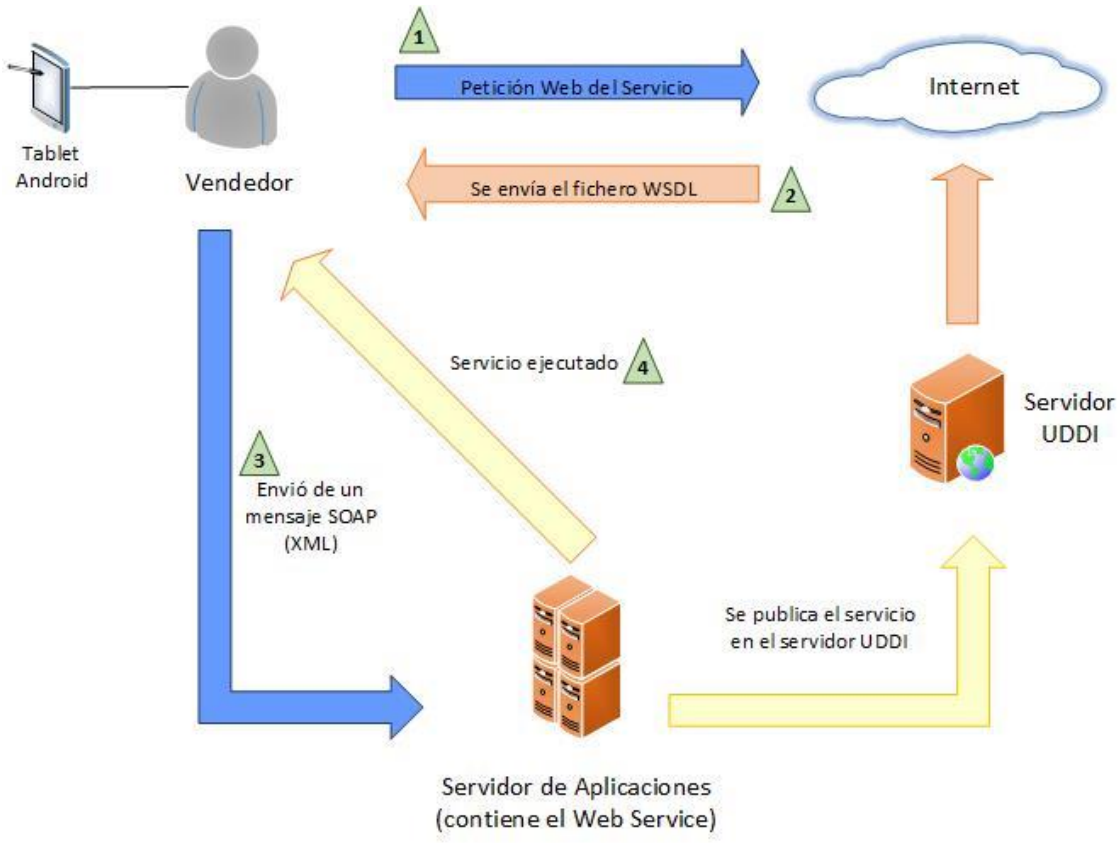


Ilustración 80. Esquema de un web Services. **Fuente:** El autor.

El vendedor accederá a la aplicación WISE y emitirá una petición web al servidor de aplicaciones en el cual residen los datos, dicho servidor deberá buscar el servicio requerido y agregar una descripción UDDI para poder publicar el servicio a través de internet.

A su vez el proveedor UDDI asociará los datos envidos con el servicio solicitado y devolverá al usuario un fichero WSDL con toda la información del web Services para luego ser completado con un mensaje SOAP (XML).

Una vez que el usuario disponga de todos los datos, se enviará al servidor de aplicaciones el mensaje SOAP completo, dicho servidor ejecutará el web Services según los parámetros que le pasaron y devolverá una respuesta al vendedor.

4.3.1 Web Services SOAP en Android

Como primer paso para poder crear un servicio web con soporte SOAP debemos considerar los requisitos previos para su creación, estos son:

Ilustración 81. Requisitos del servidor para crear el Web Services. **Fuente:** El autor.

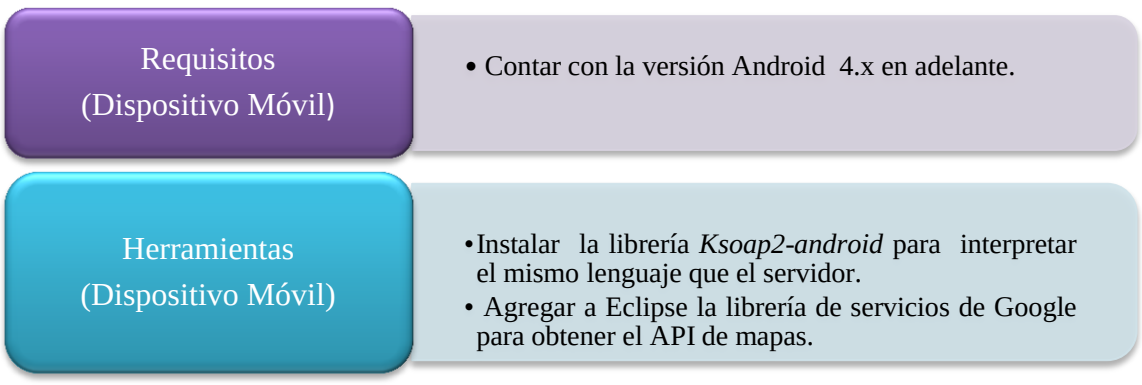


Ilustración 82. Requisitos en el dispositivo móvil para la integración del web Services. **Fuente:** El autor.

4.3.1.1 Creación de formulario de Web Services

1. En la barra de tareas de Visual Studio dar clic derecho → *Agregar nuevo elemento y elegir la opción Servicio Web.*

2. *Editar el nombre del servicio web,* en nuestro caso se llamará "*ServicioWiseMovil"*.

3. El siguiente paso consiste en la implementación del código del servicio web, debemos considerar que un servicio está compuesto por dos tipos archivos:

   - **ServicioWiseMovil.asmx:** Constituye el archivo fuente del web Services, este archivo se ejecuta en el lado del cliente.
   - **ServicioWiseMovil.vb:** Contiene el código para un método del servicio web, se ejecuta en el lado del servidor.

4. A continuación se incluye el código que contendrá la clase ServicioWiseMovil, detallando los métodos que esta clase expondrá desde el Web Services.

```
' <System.Web.Script.Services.ScriptService()> _
<WebService(Namespace:="http://servicio.wise.org/")> _
<WebServiceBinding(ConformsTo:=WsiProfiles.BasicProfile1_1)> _
<Global.Microsoft.VisualBasic.CompilerServices.DesignerGenerated()> _
Public Class ServicioWiseMovil
    Inherits System.Web.Services.WebService
    Dim Sistec As New Sistec

    <WebMethod()> Public Function GetConexion(conectado As String) As String
        Return "1"
    End Function

    <WebMethod()> Public Function GetEmpleadoActivo(usuario As String, password As String) As String
        Dim res As String = ""
        Dim cadena As String
        Dim tblDatoEmpleado As New Data.DataTable
        cadena = "select codigoEmpleado as codigo "
        cadena = cadena + " from empleados "
        cadena = cadena + " where NombreUsuarioEmpleado='" + usuario + "'"
        cadena = cadena + " and ClaveUsuarioEmpleado='" + password + "'"
        cadena = cadena + " and  rolPagoEmpleado ='1'"
        Sistec.TraerDatos(cadena, tblDatoEmpleado)
        If tblDatoEmpleado.Rows.Count = 1 Then
            res = "1"
        Else
            res = "0"
        End If
        Return res
    End Function

    Public Function GetEmpleadoEquipo(usuario As String, password As String, mac As String) As DataTable ...
```

Ilustración 83.Código fuente de la cadena de conexión creado en el web Services.
Fuente: El autor.

4.3.1.2 Consumir Web Services SOAP en Android

Paso 1: Incluir la librería ksoap2 Android a nuestro proyecto en Eclipse. Para esto debemos seguir los pasos de la Ilustración 84.

Java Build Path ⟶ *Libraries* ⟶ *Add Library* ⟶ *Agregar la librería ksoap2-android*

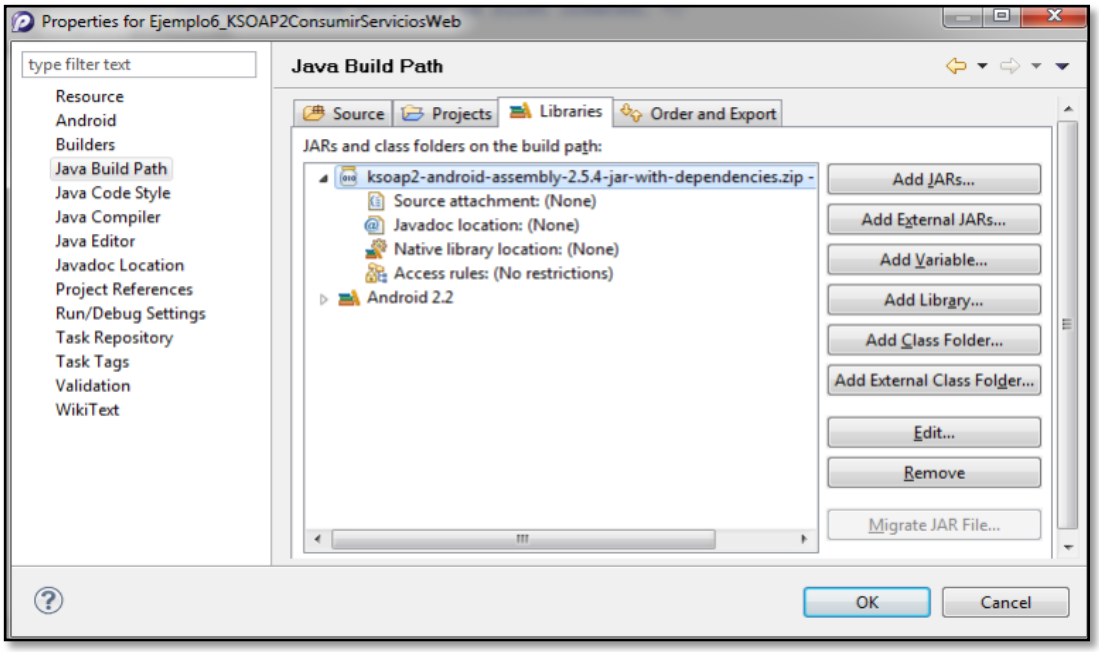


Ilustración 84.Cómo agregar la librería ksoap2-android. **Fuente:** El autor.

Paso 2: Crear la interfaz Android

main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/hello"
    />
<Button android:id="@+id/btn"
    android:text="@string/Boton"
    android:layout_height="wrap_content"
    android:layout_width="match_parent" />

<ListView android:id="@+id/lst"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content" />

</LinearLayout>
```

Ilustración 85.Cómo crear una interfaz en Android. **Fuente:** El autor.

Paso 3: Importar todos los paquetes necesarios para consumir el Web Services.

```
import java.io.IOException;
import java.lang.reflect.Type;
import java.util.ArrayList;

import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.ListView;

import org.ksoap2.SoapEnvelope;
import org.ksoap2.serialization.SoapObject;
import org.ksoap2.serialization.SoapPrimitive;
import org.ksoap2.serialization.SoapSerializationEnvelope;
import org.ksoap2.transport.HttpTransportSE;
import org.xmlpull.v1.XmlPullParserException;

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

Ilustración 86. Paquetes necesarios para consumir el Web Services. **Fuente:** El autor.

Paso 4: Procedemos a declarar la variables a nivel de clase y crear el código que se encargara de invocar al web Services. Ilustración 87

```
//Constantes para la invocacion del web service
private static final String NAMESPACE = "http://servicio.wise.org/";
private String  URL="http://190.154.xxx.xxx/ServicioWiseMovil.asmx";
private static final String METHOD_NAME_CON = "GetConexion";
private static final String SOAP_ACTION_CON ="http://servicio.wise.org/"+METHOD_NAME_CON+"";

public String getConexion(String valor){
    String res=null;
    //Se crea un objeto de tipo SoapObjecto. Permite hacer el llamado al WS
    resultsRequestSOAP = new SoapObject(NAMESPACE, METHOD_NAME_CON);
    resultsRequestSOAP.addProperty("conectado", valor); // Paso parametros al WS
    //Para obtener informacion del WS , se puede consultar http://192.168.10.3/ServicioWiseMovil.asmx?WSDL
    envelope = new SoapSerializationEnvelope(SoapEnvelope.VER11);
    envelope.bodyOut = resultsRequestSOAP;
    //Se establece que el servicio web esta hacho en .net
    envelope.dotNet = true;
    envelope.encodingStyle = SoapSerializationEnvelope.XSD;
    //Para acceder al WS se crea un objeto de tipo HttpTransportSE , esto es propio de la libreia KSoap
    HttpTransportSE androidHttpTransport= null;
    try {
        androidHttpTransport = new HttpTransportSE(URL);
        androidHttpTransport.debug = true;
        //Llamado al servicio web . Son el nombre del SoapAction, que se encuentra en la documentacion del servicio web y el objeto envelope
        androidHttpTransport.call(SOAP_ACTION_CON, envelope);
        //Respuesta del Servicio web
        SoapPrimitive resSoap =(SoapPrimitive)envelope.getResponse();
        res=resSoap.toString();
        if(res.equals("1")){
            res="1";
        }else{
            res="0";
        }
      }catch (Exception e){
        e.printStackTrace();
          res="0";
    }
    return res;
}
```

Ilustración 87.Código para invocar al Web Services**. Fuente:** El autor.

Paso 5: Para terminar debemos agregar el permiso de internet modificando el archivo Manifest.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="ec.com.sistec"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="14"
        android:targetSdkVersion="21" />

    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.READ_LOGS" />
    <uses-permission android:name="android.permission.READ_PHONE_STATE" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED" />
    <uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
    <uses-permission android:name="android.permission.ACCESS_MOCK_LOCATION" />
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
    <uses-permission android:name="android.permission.ACCESS_LOCATION_EXTRA_COMMANDS" />

    <permission
        android:name="org.example.ejemplogooglemaps.permission.MAPS_RECEIVE"
        android:protectionLevel="signature" />

    <uses-permission android:name="org.example.ejemplogooglemaps.permission.MAPS_RECEIVE" />
    <uses-permission android:name="com.google.android.providers.gsf.permission.READ_GSERVICES" />

    <uses-feature
        android:glEsVersion="0x00020000"
        android:required="true" />

    <uses-permission android:name="android.permission.CAMERA" />


    <application
```

Ilustración 88.Cómo agregar el permiso de internet al Web Services**. Fuente:** El autor.

4.3.1.3 Seguridad del Web Services

Se implementó un algoritmo alfanumérico personalizado dado que disponía de dos sistemas que funcionaban en diferentes plataformas. Este algoritmo permite cifrar y descifrar la información enviada por el vendedor, el proceso que se sigue es encriptar el nombre de usuario y contraseña creando una clave única para este conjunto de datos. Una vez que el mensaje ha sido encriptado será comparado con los datos del servidor central, permitiendo solo al administrador conocer el algoritmo de desencriptación. El objetivo de aplicar un algoritmo personalizado es agregar un nivel mayor de dificultad para descifrar la cadena original, permitiendo transferir la información de la empresa a través de un medio inseguro sin comprometer la integridad de los datos.

4.4 Paquetes y recursos para desarrollar aplicaciones Android sin conexión a internet

En las redes móviles en donde los nodos cambian constantemente, donde no hay apoyo a la infraestructura y sobre todo donde el usuario cree que todo lo que el dispositivo hace depende de la red, crear aplicaciones fuera de línea representa disponibilidad en cualquier lugar, tiempo o circunstancia.

Seguir creando aplicaciones con la mentalidad de modo online no es factible ya que los posibles escenarios de desconexión son mayores tanto por cobertura, ancho de banda, factores externos, etc. [57]. Para empezar a desarrollar una aplicación sin conexión necesitamos prescindir de algunas herramientas y librerías que facilitan el trabajo de desarrollo, creación y depuración de código en Android, éstas son:

- **Java JDK:** Comprende un conjunto de librerías gratuitas que permiten desarrollar, copilar y ejecutar programas creados en Java.

- **Eclipse:** Representa nuestro entorno de trabajo, está compuesto por varias herramientas de código abierto y multiplataforma.

- **ADT Android:** Conjunto de herramientas de desarrollo para creación de aplicaciones Android.[59]

- **SDK Android:** Kit de desarrollo para Android, posee un emulador con las diferentes versiones del sistema operativo [59].

4.4.1 Instalación de JDK 8

1) Para Windows debemos descargar la versión jdk 8 del sitio oficial de Oracle, esta última versión ha incorporado interesantes novedades en cuanto a soporte para lenguajes dinámicos, actualización de seguridad, internacionalización, manejo de estándares, etc.

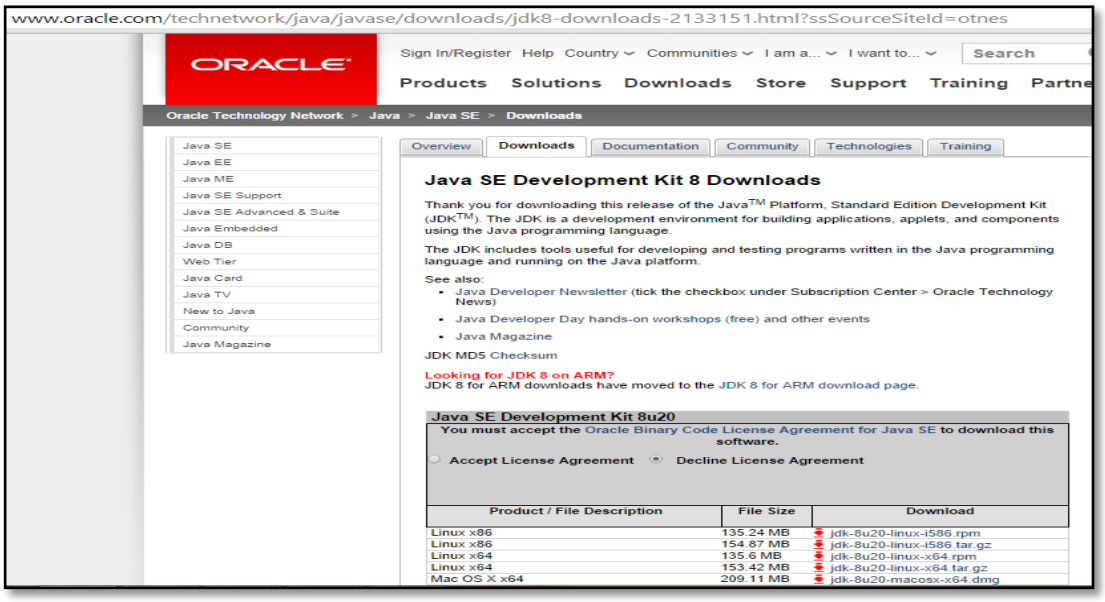


Ilustración 89. Sitio Web Oficial de Oracle para descargar la versión JDK 8 [58].

2) Procedemos a aceptar los términos de licenciamiento para luego ir a la sección de Java SE Development Kit 8u20 y descargar la versión de jdk8 para Windowsx64.

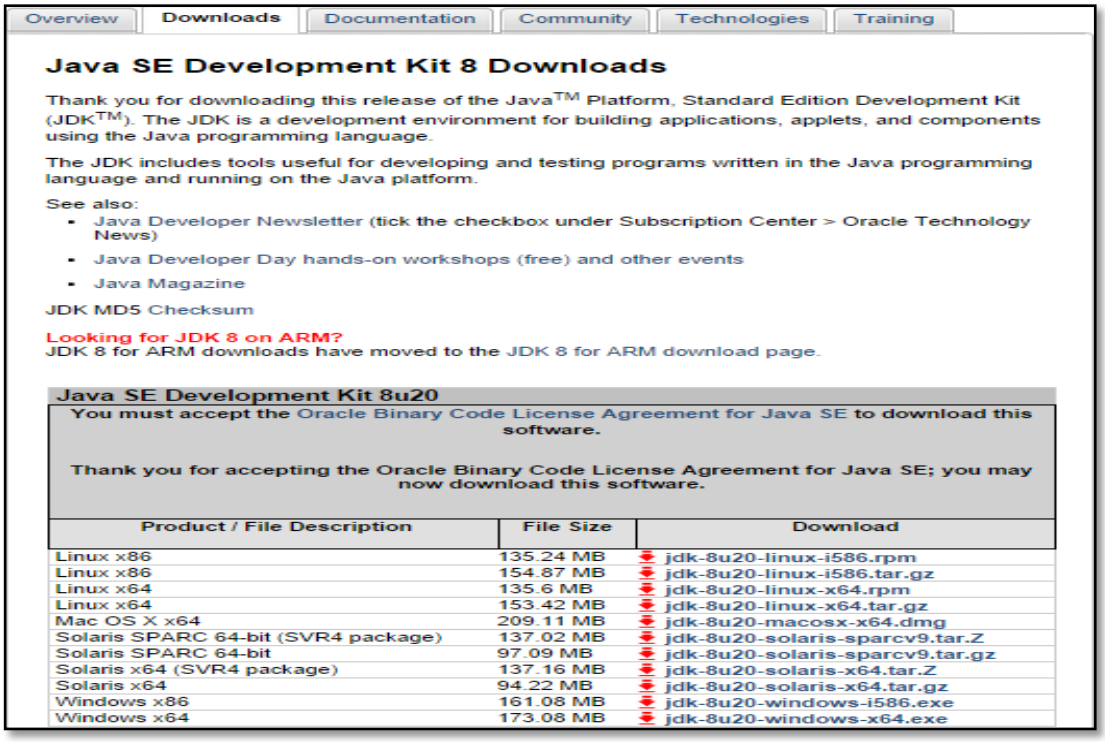


Ilustración 90. Descarga del JDK8 [58].

3) Para iniciar la instalación damos doble clic en el archivo ejecutable descargado y posteriormente clic en siguiente.

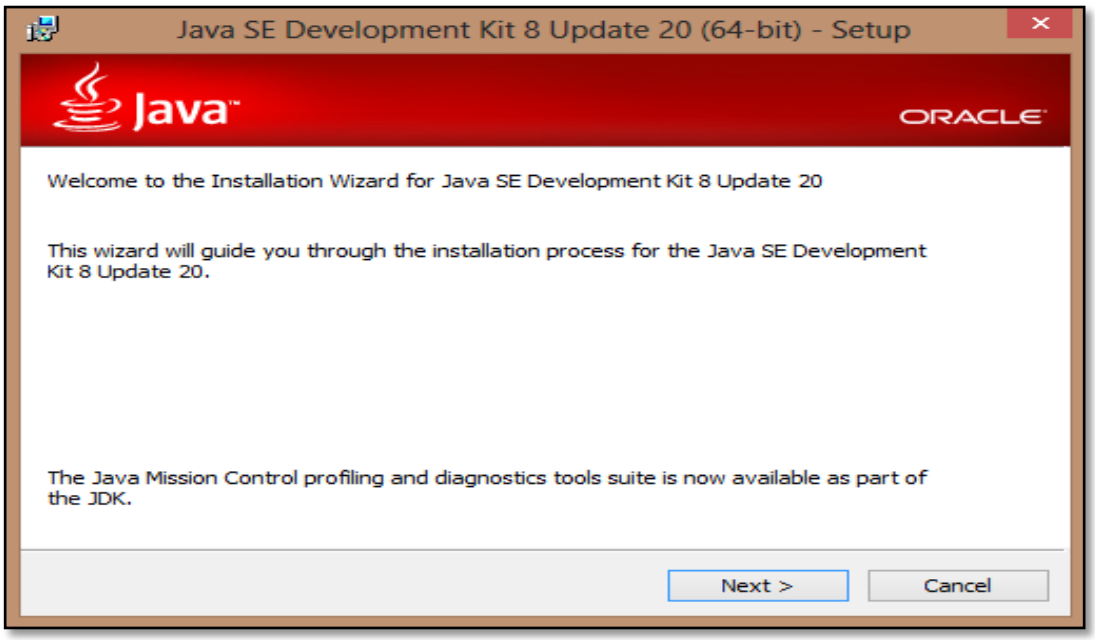


Ilustración 91. Instalación de JDK 8. **Fuente:** El autor.

4) En la siguiente pantalla debemos escoger las características que deseamos instalar y dar clic en siguiente.

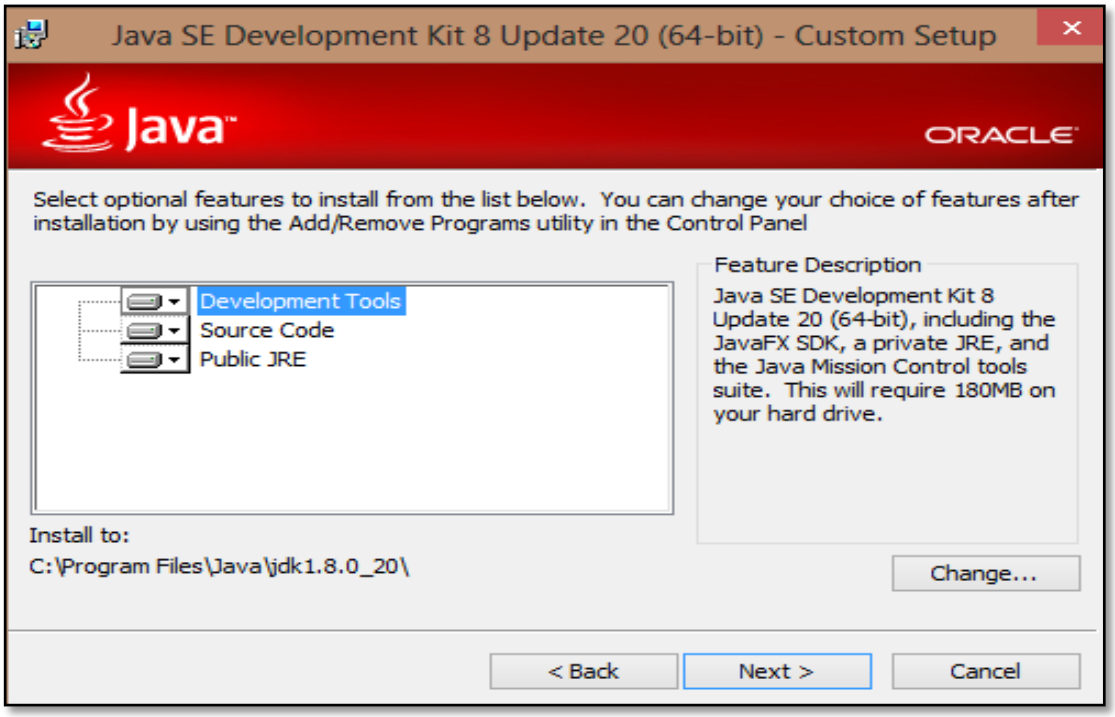


Ilustración 92. Selección de las características de instalación. **Fuente:** El autor.

5) Una vez que la instalación se ha completado satisfactoriamente, aparecerá la siguiente pantalla, aquí damos click en *Close* para finalizar la instalación.

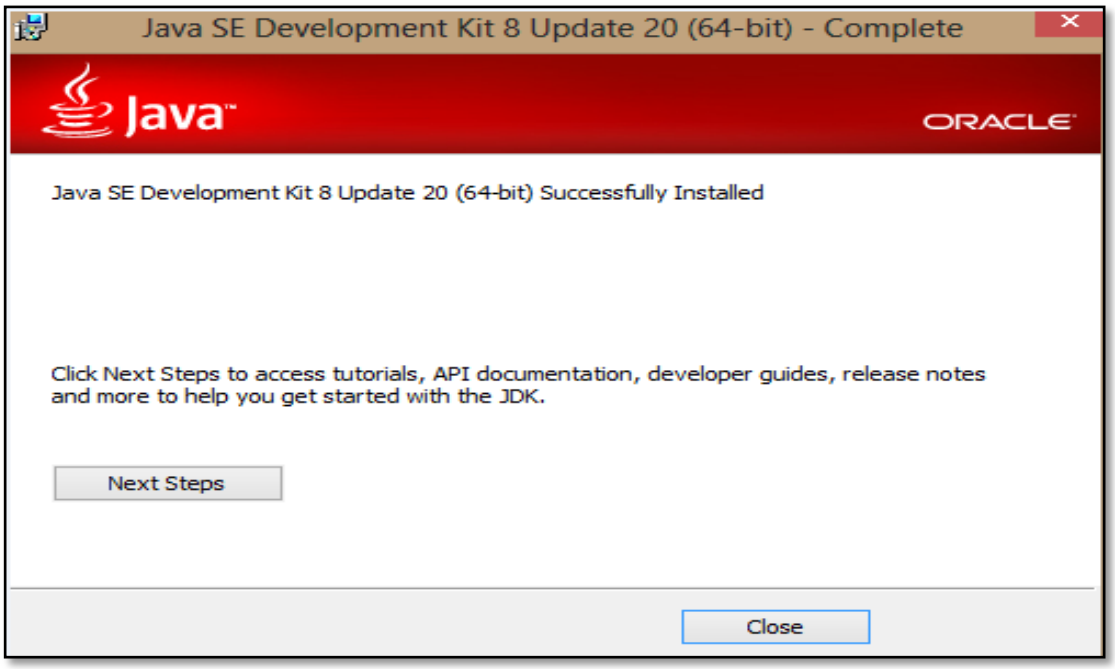


Ilustración 93. Selección de las características de instalación. **Fuente:** El autor.

4.4.2 Instalación de SDK + Eclipse + ADT de Android

Actualmente el Kit desarrollo de Android incluye una variedad de componentes para agilizar el desarrollo de aplicaciones Android, el kit incluye:

- ✓ **Eclipse + ADT (Android Development Tools):** Incluye la versión de eclipse June y el plugin para el IDE de Eclipse.

- ✓ **Android SDK Tools:** Incluye el conjunto completo de herramientas de desarrollo y depuración para el SDK de Android.

- ✓ **Android Platform-tools:** Contiene las diferentes versiones de Android que podrán ser instaladas automáticamente.

GUIA DE INSTALACIÓN

1) Acceder al sitio oficial Android Developer y descargar el Kit de desarrollo de Android, para esto damos click en *Download Eclipse ADT with the Android SDK for Windows.*

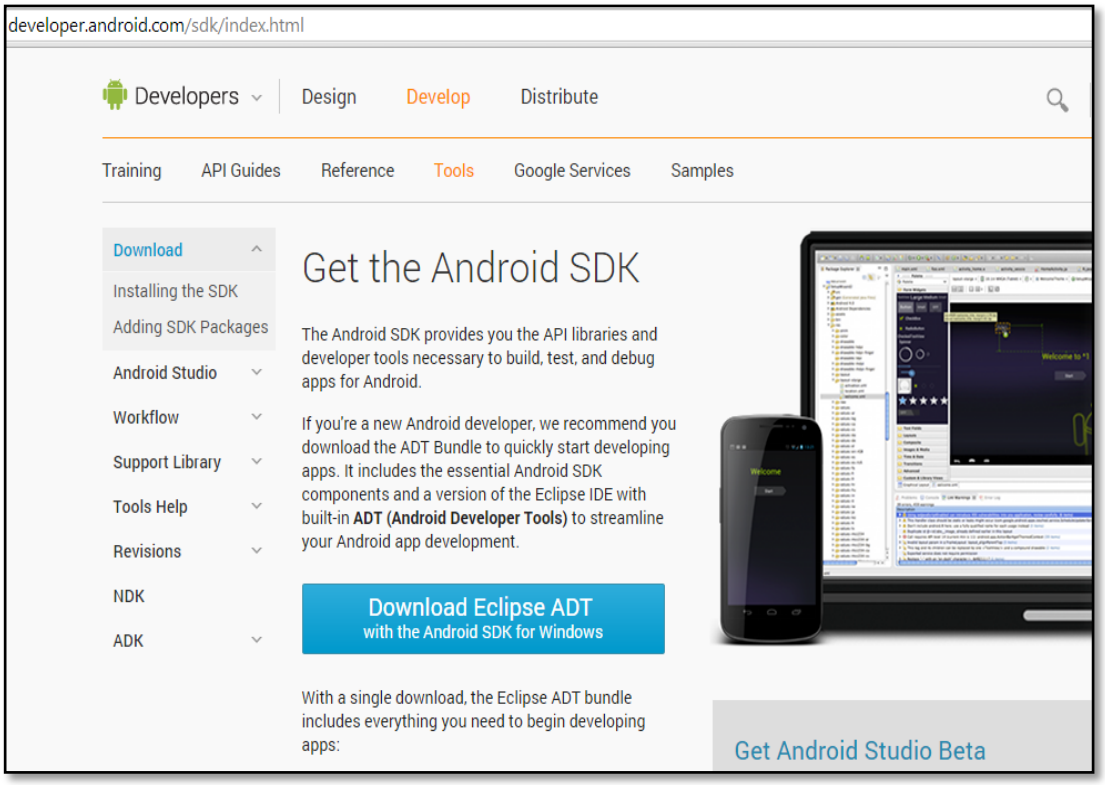


Ilustración 94.Página oficial de Android para descargar el Kit de Desarrollo (SDK) [59].

2) Una vez descargado el archivo ***adt-bundle-windows-x86_64-2014702.zip*** tenemos que descomprimirlo en una carpeta creada en el disco C. Para luego ejecutar el archivo *SDK Manager.exe.*

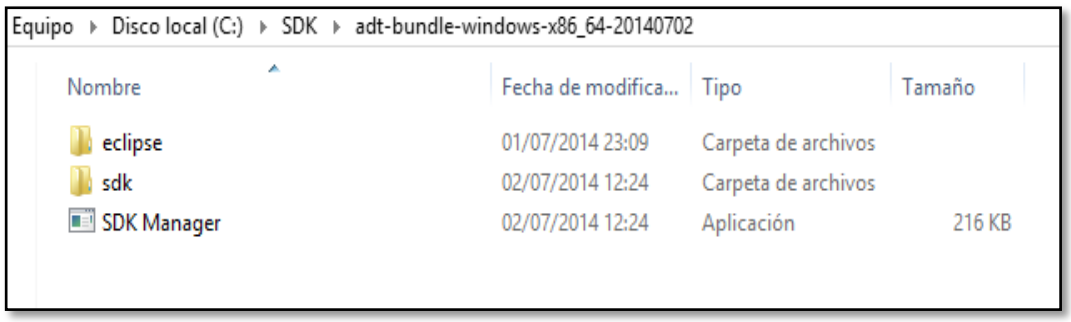


Ilustración 95. Ubicación del archivo SDK. **Fuente:** El autor.

3) Seleccionamos todas las *Tools*, todo el *Android 4.3.1 (API18)* y todos los *Extras* para luego dar click en el botón *Install packages.*

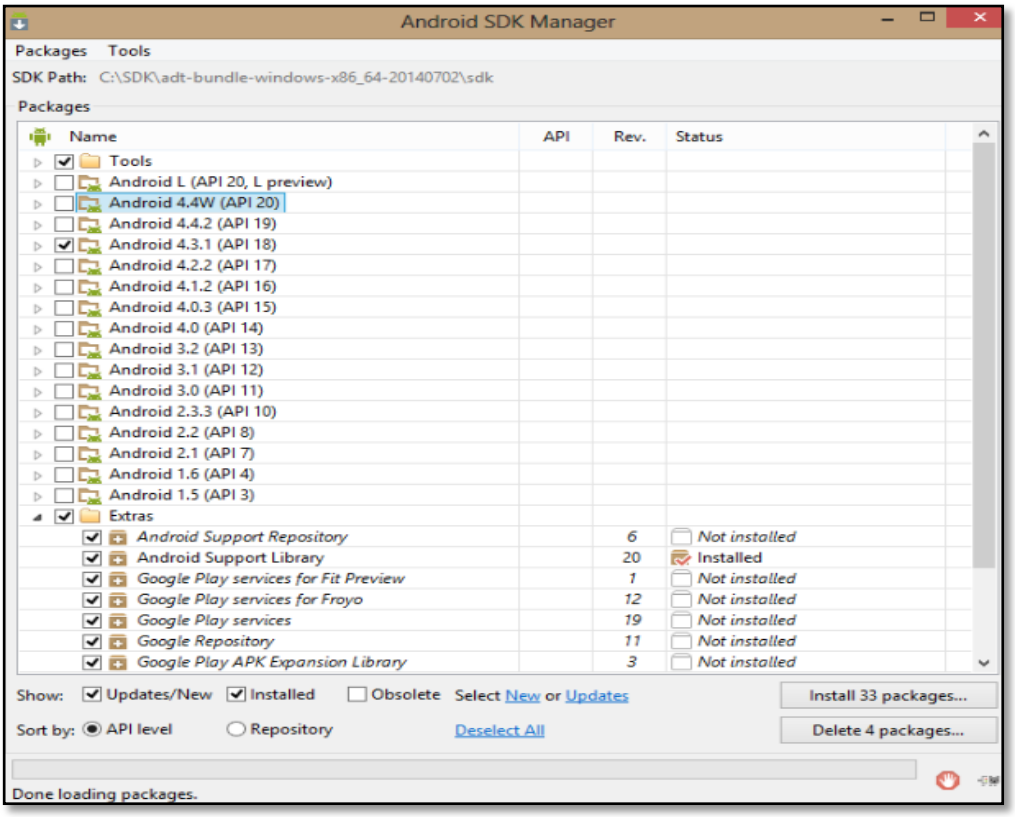


Ilustración 96. Selección del kit de herramientas de Android. **Fuente:** El autor.

4) Aceptamos los términos de licenciamiento y damos click en *Install* para finalizar.

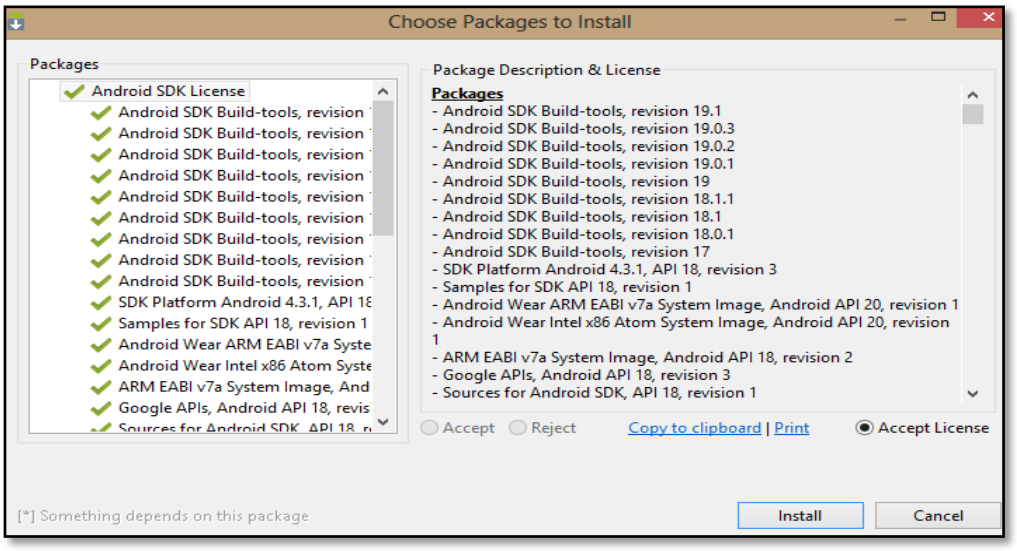


Ilustración 97. Instalación del kit de desarrollo de Android. **Fuente:** El autor.

4.4.3 Creación de un primer proyecto

1. Vamos a la carpeta inicial en el disco C y buscamos la opción de *Eclipse* para ejecutarlo. Al abrir eclipse por primera vez nos pedirá escoger el wokspace o ruta de trabajo. Aceptamos la ruta que nos sugiere.

2. Para crear un nuevo proyecto vamos a File>New>Project. Si el plugin de Android se ha instalado correctamente el cuadro de diálogo debe tener la opción *Android Application Project.*

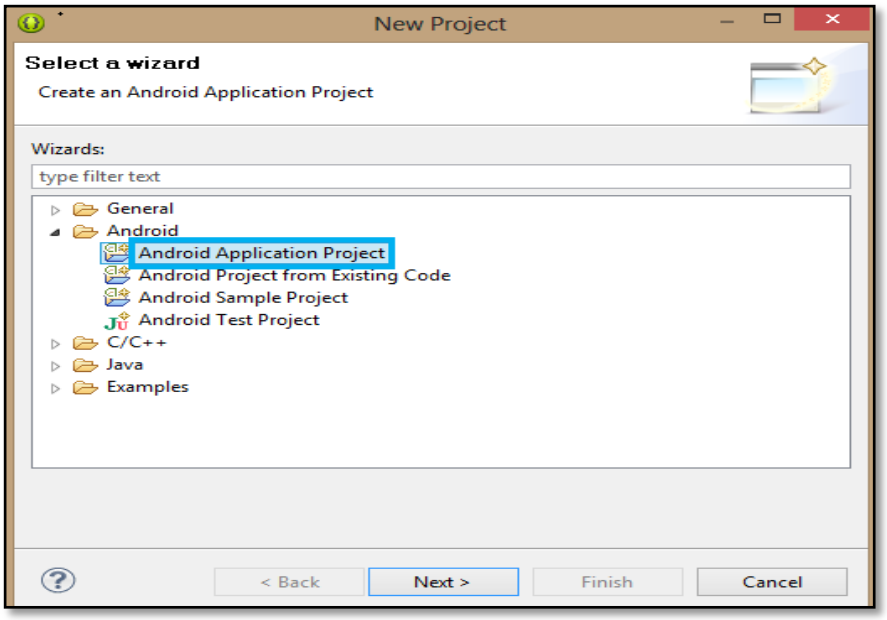


Ilustración 98. Creación de un nuevo proyecto en Android. **Fuente:** El autor.

3. Llenamos los siguientes campos para configurar el nuevo proyecto:
   - **Aplication Name:** Nombre que se le dará a la aplicación.
   - **ProjectName:** Representa el nombre del proyecto Android.
   - **Package Name:** Es el nombre del paquete sobre el cual será desarrollado el proyecto.
   - **Minium Required SDK:** La versión mínima sobre la cual podrá correr la aplicación.
   - **Target SDK:** Versión SDK sobre la cual correrá la aplicación.
   - **Compile With:** Determina la versión sobre la cual copilara el programa.
   - **Theme:** Define el estilo de la interfaz gráfica del usuario.

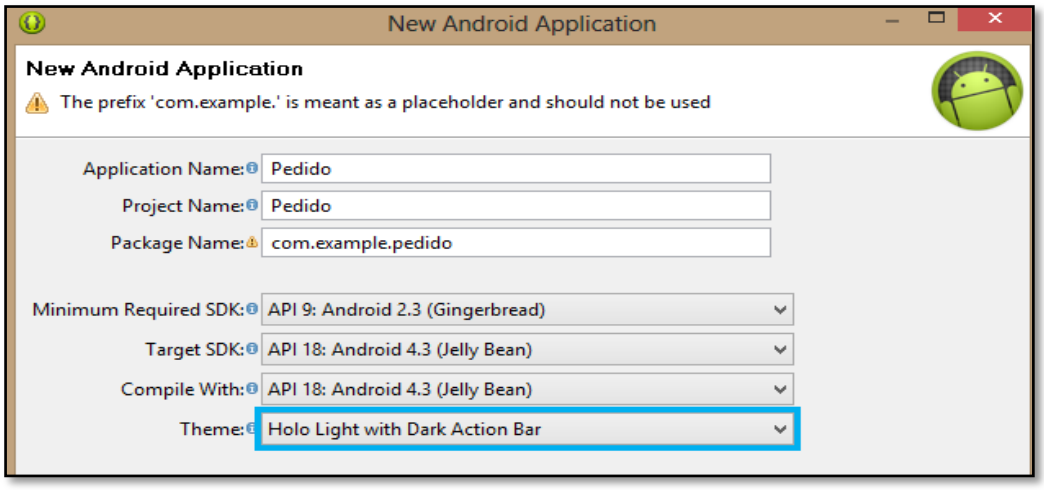


Ilustración 99. Configuración de un nuevo proyecto Android. **Fuente:** El autor.

4. En la siguiente pantalla que nos aparece, debemos seleccionar el icono que queremos para nuestra aplicación.

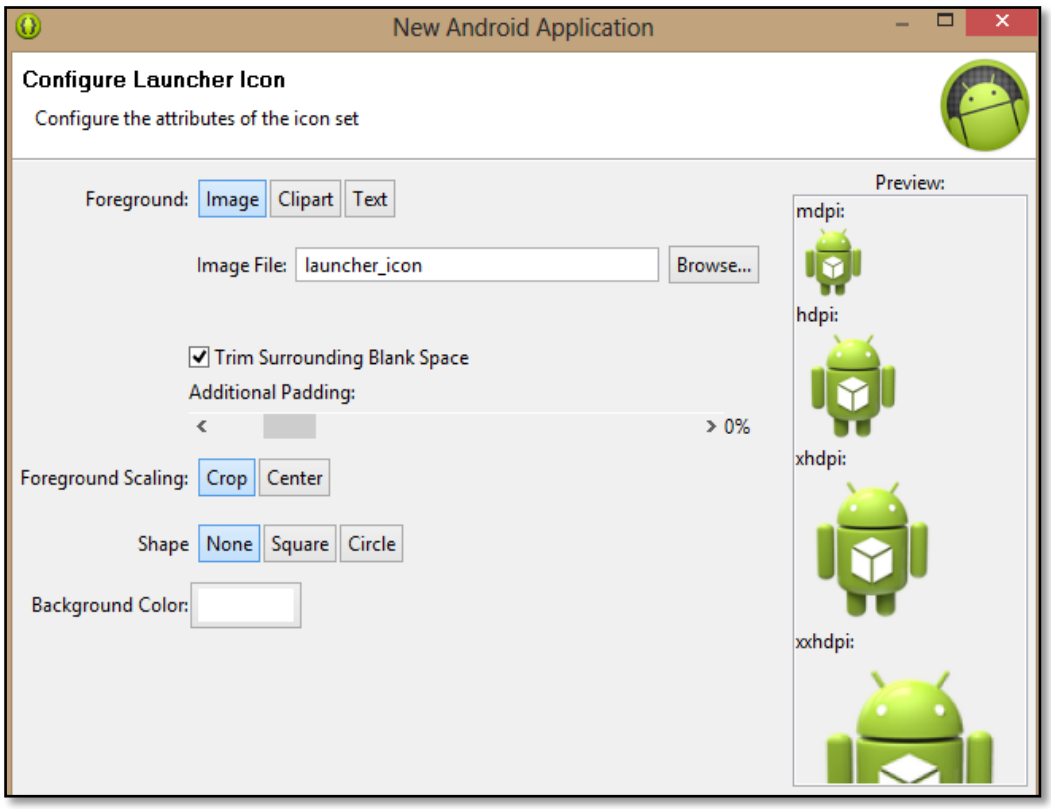


Ilustración 100. Configuración del icono de Android. **Fuente:** El autor.

5. Procedemos a crear una actividad, para esto seleccionamos la opción *Blank Activit.*

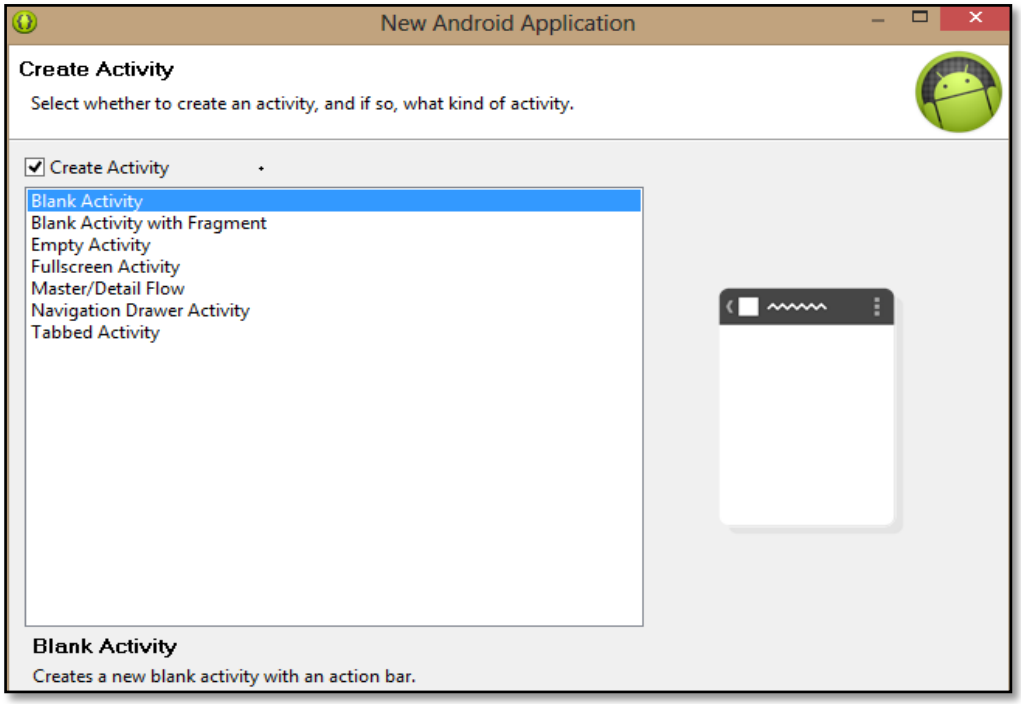


Ilustración 101. Creación de una actividad en Android. **Fuente:** El autor.

6. Llenamos la siguiente información y pulsamos en *Finish* para terminar.

- ✓ **Activity Name:** Representa la clase principal encargada de la ejecución del programa.
- ✓ **LayoutName:** Es el nombre que le vamos a dar al fichero XML que va a contener la vista de nuestra primera pantalla.
- ✓

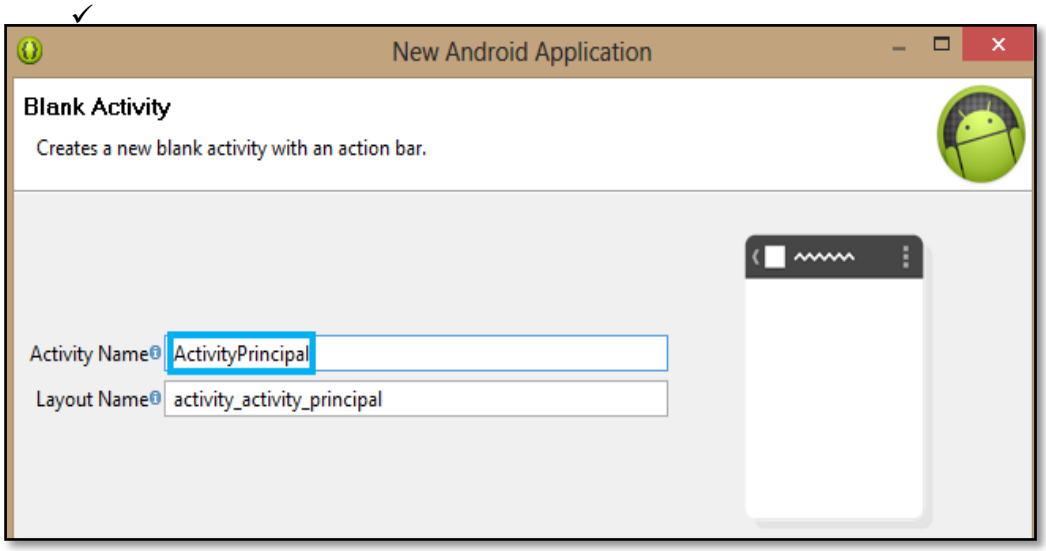


Ilustración 102. Configuración de una actividad en Android. **Fuente:** El autor.

7. Para ejecutar el proyecto debemos dar click en Run>Run As>Android Application

Ilustración 103. Ejecución de un proyecto Android. **Fuente:** El autor.

4.4.4 Funcionamiento de la aplicación móvil offline

La aplicación cuando se instala se descarga toda la información necesaria para el funcionamiento, los datos se descargan del servidor central porque el sistema principal WISE ya está implementado, la aplicación es una adaptación al sistema mencionado. Descargar los datos es necesario para la función offline que va cumplir la aplicación. La aplicación debe tener los mismos datos del servidor central, según el esquema que se diseñó para realizar las órdenes de pedido móvil.

En la aplicación se puede registrar localmente las órdenes de pedidos, y no necesita estar conectado al internet, y cuando existe la conexión existe un servicio donde está ejecutando en segundo plano donde detecta la comunicación al servidor mediante el internet para que pueda enviar y reflejar en el servidor central, para que proceda el proceso de despacho de la misma.

CAPÍTULO 5

5. EJECUCIÓN DE PRUEBAS

A medida que se desarrolla una aplicación, es necesario comprobar si satisface su especificación y si cumple con la funcionalidad para la cual fue pensado. Proceso comúnmente conocido como verificación y validación (V&V).

Las pruebas tienen como propósito llevar a cabo un proceso de retroalimentación de los usuarios, de tal forma que, tras la observación de su comportamiento, sea posible corregir y mejorar aspectos de usabilidad y funcionalidad. Para la V&V de la aplicación móvil se ha definido dos fases de pruebas denominadas: pruebas de usabilidad y pruebas de funcionalidad las mismas que serán detalladas a continuación.

5.1 Ejecución de la prueba de usabilidad

Este tipo de pruebas puede ser aplicado antes de tener una aplicación terminada, el objetivo es detectar errores de diseño y desarrollo en etapas tempranas. El proceso de familiarización con la aplicación arrancó con la instalación de la versión prueba en un solo equipo, en las siguientes semanas se procedió a instalar la aplicación mejorada en 3 equipos más hasta contar actualmente con 7 vendedores que disponen la última versión.

- ***Grupo A (Vendedores)***

EMPRESA	***VENDEDORES***			
	Nombres	**Apellidos**	**Profesión**	**Edad**
Agrota Cia. Ltda.	Juan Pablo	León Avecillas	Ing. Agrónomo	36
Agrota Cia. Ltda.	Kleber Vinicio	Peralta Fonseca	Asesor Comercial	32
Agrota Cia. Ltda.	Christian Eduardo	Armas Mugmal	Ing. Agropecuario	30
Agrota Cia. Ltda.	Roberto Marco	Chang Angulo	Asesor Comercial	36
Agrota Cia. Ltda.	Carlos Santiago	Torres Robles	Consultor	28
Agrota Cia. Ltda.	Yuri Simón	Farías Soledispa	Ing. Agrónomo	32
Agrota Cia. Ltda.	Alejandro Enrique	Calle Roca	Ing. Agrónomo	30

Tabla 40. Participantes del test de usabilidad. **Fuente:** El autor.

El proceso de socialización con los vendedores estuvo a cargo de los desarrolladores y el equipo de supervisión de la empresa. Aquí se expuso las diferentes ideas y mejoras de los vendedores que ya estaban usando la aplicación en su trabajo diario. Al crear una lluvia de ideas pudimos constatar la gran aceptación e interés por parte de los usuarios.

Ilustración 104. Prueba de usabilidad con los vendedores de Agrota. **Fuente:** El autor.

5.2 Ejecución de la prueba de funcionalidad

La prueba de caja blanca es un método de diseño de casos de prueba, su objetivo es comprobar la lógica del programa examinando la estructura del código fuente, la probabilidad de fallos en el código, el esfuerzo necesario para comprobar todos los caminos posibles, etc.

También se evaluará la complejidad ciclomática, con esto nos referimos a que mientras más compleja sea la lógica del código, más difícil será de entender, mantener y probar. Se establecerá un rango de referencia para medir la complejidad ciclomática tal como se puede apreciar en la siguiente tabla [60,61].

VALOR	*METODO*	*RIESGO*
<=10	Método sencillo	Sin mucho riesgo
>10, <=20	Método medianamente complejo	Con riesgo moderado
>20,<=50	Método inestable	Altísimo riesgo

Tabla 41. Valor de referencia para la complejidad ciclomática [61].

El criterio que se seguirá para evaluar el código fuente de la aplicación móvil será a partir de un valor de referencia de 20 tomado como máxima complejidad ciclomática aceptable para un método, si el código llegara a superar el valor se deberá idear un método de optimización de código.

- ***Prueba de Caja Blanca***

Consiste en analizar la complegidad del flujo del programa desglosandolo en un conjunto de caminos posibles,de esta forma sera mucho mas fácil garantizar que todas las condiciones se ejecutan correctamente.

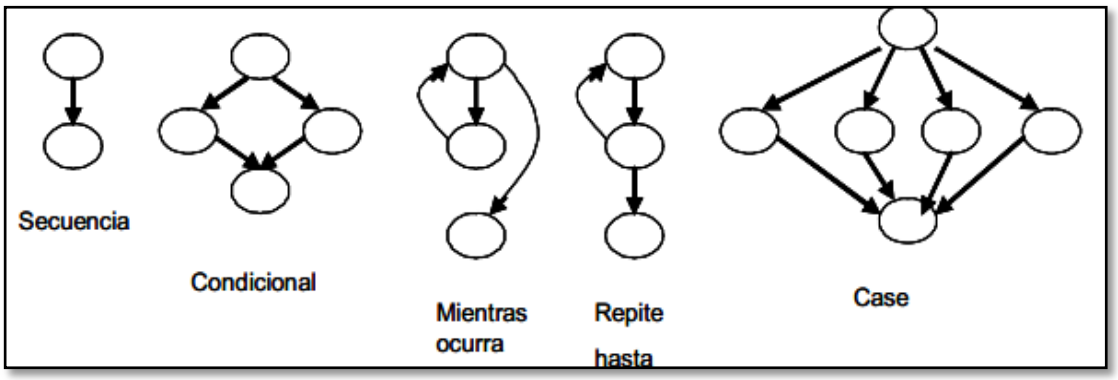


Ilustración 105. Simbología del camino básico [62].

a) Prueba de Caja Blanca 001:Ingresar al Sistema

```
    private OnClickListener btnIngresoUsuario_click = new OnClickListener() {
        @Override
        public void onClick(View v) {
            // TODO Auto-generated method stub
            ArrayList<Parametro> listarParametros=gParametro.getParametros("");;//Lista lor parametros que se grabo, al momento de instalar la aplicacion como la: ip,numero de
 1.  if(listarParametros.size()>0){datosTablaParametro();
 2.      if(vereficarNumeroIntentos()){// Verifica el numero de intentos asignado en la tabla parametros,  con el contador si es igual  devuelve true
 3.          //Si es true bloquea en ingreso con el tiempo asingando como norma
 4.      }else{//caso contrario sigue intentando
 5.          if(txtUsuario.getText().toString().trim().equals("") || txtClave.getText().toString().trim().equals("")){//Verifica si los campos de usuario y contraseña si es
 6.              Toast.makeText(getApplicationContext(),"Mensaje:Ingrese usuario y/o contraseña",Toast.LENGTH_LONG).show();
 7.          }else{
 8.              if(verificaConexionInternet(LoginActivity.this)){//Verifica si existe la conexion a internet si es true
 9.                  progressDialog= ProgressDialog.show(LoginActivity.this, "Por favor Espere ...", "Verificando si usuario esta activo...", true);
                     new ConexionServerTask().execute("2");//La clase de hilo  que se ejecuta en segundo plano para conexion al servidor central y
                                                           //  comparar los datos del usuario //si devuelve 1 sera correcto caso contrario devuelve mensaje de aviso
10.              }else{
                     //Si el usuario no tiene conexion y ya esta registrado en el dispositivo podra ingresar localmente
                     ArrayList<Empleado> listarEmpleados=gEmpleados.getEmpleadoAll("");
11.                  if(listarEmpleados.size()>0){//Verifica si existe datos del empleado o usuario
                         Log.d("Numero de Empleados ",""+listarEmpleados.size());
12.                      Empleado listarEmpleado=gEmpleados.validarUsuarioGetEmpleado(txtUsuario.getText().toString().trim(),txtClave.getText().toString().trim());//Verific
                                                                                                                                  //es que existe datos del usuarioS
13.                      if(listarEmpleado!=null){//Si existe
                             //ingresa o logue a la aplicacion
                             codigoEmpleado=listarEmpleado.getCodigo();
14.                          ingresarAplicacion();
15.                      }else{mensajeDeAvisoUsuario();//Devuelve un mensaje
                         }
16.                  }else{
                         Toast.makeText(getApplicationContext(),"Mensaje:No existe usuarios para el ingreso",Toast.LENGTH_LONG).show();
                     }
                 }}}
17.  }else{//Si no existe los datos de la tabla parametro, no puede conectarse por lo tanto no tiene ip para la conexion
             mostrarDialogoIngresarIp();//Ingresar el ip para la conexion respectiva
         }
        contador++;//cuenta numero de intentos
        }
18. }
```

Ilustración 106. Código para validar las entradas del Ingreso al sistema. **Fuente:** El autor.

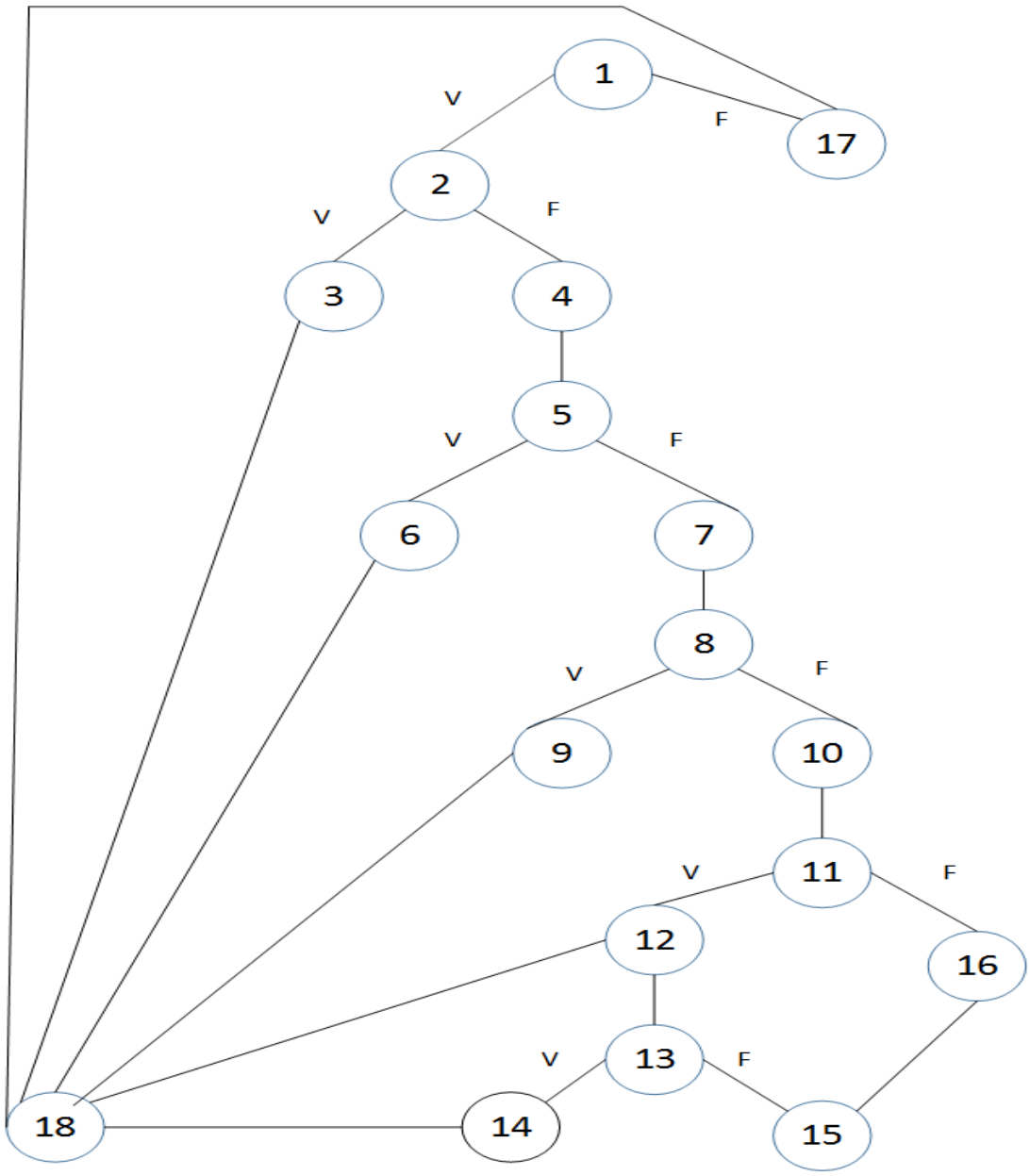


Ilustración 107. Grafo del camino principal para el Ingreso al Sistema. **Fuente:** El autor.

PRUEBA DE CAJA BLANCA-001: Ingresar al Sistema		
Fórmula: **V(G) = P + 1 =6 + 1 = 7**	***Número de caminos***	***Ruta de caminos independientes***
Donde: **V(G):** Complejidad ciclomática **P:**Nodos Predicados =6	**1**	1 – 17 – 18
	2	1 – 2 – 3 - 18
	3	1 – 2 – 4 – 5 – 6 - 18
	4	1 – 2 – 4 – 5 -7 – 8 – 9 - 18
	5	1 – 2 – 4 – 5 – 7 – 8 -10 – 11 - 12 – 18
	6	1 – 2 – 4 – 5 – 7 – 8 – 10 – 11 – 12 – 13 – 14 – 18
	7	1 – 2 – 4 – 5 – 7 – 8 – 10 – 11- 16 – 15 -13 – 14 - 18

Tabla 42. Prueba de Caja Blanca-001: Número de caminos independientes. **Fuente:** El autor.

b) Prueba de Caja Blanca 004: Registrar Orden de Pedido

```
1.  public void grabarOrdenPedido(){
        String codigoOrdenCabecera=ctrl.getCod();
        double cantidadItems=0;
        String codCliente=lblCodigoCliente.getText().toString();
2.      if(codCliente!=""){//verifica si el cliente esta seleccionado que no este vacio
3.          ArrayList<OrpDetalleTemporal> listarOrpTemp=gOrpDetalleTemp.getOrpDetalleTemporal("");
4.          if(listarOrpTemp.size()>0){//verifica que item tenca cantidad para poder grabar
5.              for(int j=0;j<listarOrpTemp.size();j++){//bucle para ver si todos los items tenga cantidad mayor a cero
6.                  if(listarOrpTemp.get(j).getCantidadOrpDetalle()==0){//si el item tiene la cantidad en cero cambia el estado(cantidadItems) en 2 cual identifica q no tiene cant
                        cantidadItems=2;
                    }
                }
7.              if(cantidadItems==2){//si estado(cantidadItems) es igual a 2 mesaje
8.                  Toast.makeText(getApplicationContext(),"La cantidad de algunos de los Items esta en cero por favor verifique ",Toast.LENGTH_LONG).show();
9.              }else{
10.                 if(editTextObservacionOrp.getText().toString().trim().equals("")){//Ingresar la observacion para grabas la orden
11.                     Toast.makeText(getApplicationContext(),"Usted debe ingresas la observacion para grabar la orden  ",Toast.LENGTH_LONG).show();
12.                 }else{
                        // Ingresa a la tabla la cabecera de la orden de pedido
                        OrpCabecera cab_orp=new OrpCabecera();
                        cab_orp.setCodigoOrpCabecera(codigoOrdenCabecera);
                        gOrpCabecera.insertar(cab_orp);
                        for(int i=0;i<listarOrpTemp.size();i++){
                            OrpDetalle det_orp=new OrpDetalle();//ingresa el detalle de la orden
                            det_orp.setCodigoOrpDetalle(listarOrpTemp.get(i).getCodigoOrpDetalle());
                            det_orp.setCodigoOrpCabeceraDetalle(codigoOrdenCabecera);
                            gOrpDetalle.insertar(det_orp);
                            Toast.makeText(getApplicationContext(),"La Orden de pedido se grabo correctamente",Toast.LENGTH_LONG).show();
                        }
                    }
                }
13.         }else{
                Toast.makeText(getApplicationContext(),"La orden no contiene items por favor seleccione al menos uno",Toast.LENGTH_LONG).show();
            }
14.     }else{
15.         Toast.makeText(getApplicationContext(),"Seleccione el cliente para grabar la Orden",Toast.LENGTH_LONG).show();
        }

16. }
```

Ilustración 108.Código para validar las entradas del Registro Orden de Pedido. **Fuente:** El autor.

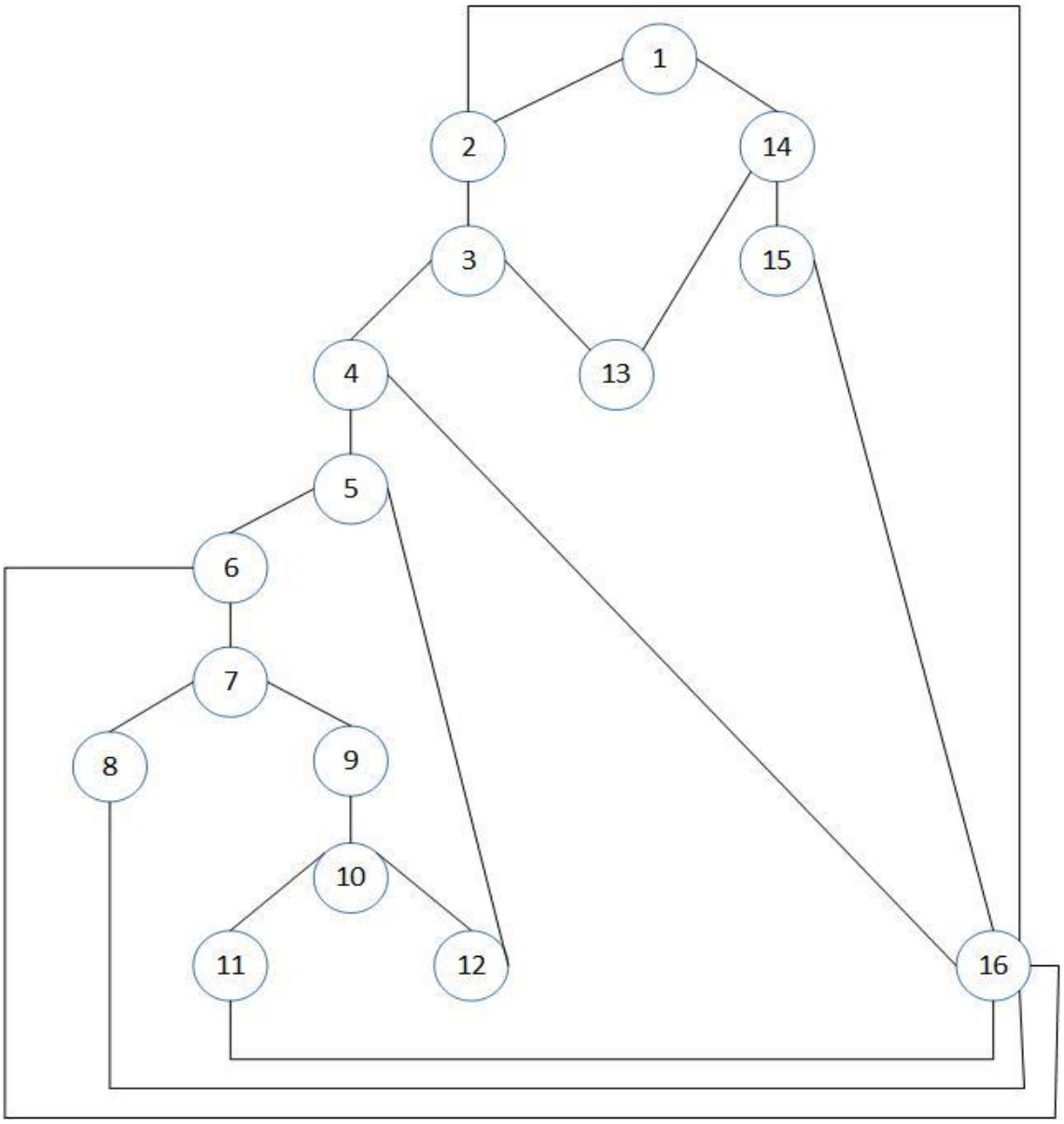


Ilustración 109. Grafo del camino principal para el Registro de Pedido [El autor].

PRUEBA DE CAJA BLANCA-004: Registrar Orden de Pedido		
Fórmula: **V(G) = P + 1 =6 + 1 = 7**	***Número de caminos***	***Ruta de caminos independientes***
Donde: **V(G):** Complejidad ciclomática **P:**Nodos Predicados =6	**1**	1 – 12 – 16
	2	1 – 2 – 3 – 13 – 14 – 15 – 16
	3	1 – 2 – 3 – 4 – 16
	4	1 – 2 – 3 – 4 – 5 – 12 – 10 – 11 – 16
	5	1 – 2 – 3 – 4 – 5 – 6 – 16
	6	1 – 2 – 3 – 4 – 5 – 6 – 7 – 9 – 10 –11 – 16
	7	1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 16

Tabla 43. Prueba de Caja Blanca-004: Número de caminos independientes. **Fuente:** El autor.

c) Prueba de Caja Blanca 006:Agregar prodcutos por catálogo a la orden de pedido

```
public void agregarItemOrden(String coidgoProducto){
1.if(lblProductoSeccionadoCod.getText().toString().trim().equals("")){//verifica el produto esta vacio para mostrar un mensaje de aviso
2. Toast.makeText(getApplicationContext(),"Seleccione producto para agregar a la orden de pedido",Toast.LENGTH_LONG).show();
3. }else{// si no esta vacio
4.if(verificarDescuentoLineaCalificacion(lblProductoSeccionadoCod.getText().toString().trim())>=0){//verifica el descuento de la linea, del producto que perteneces y po
5.productosDetalle=gProducto.getProductos(" WHERE codigoproducto ='"+lblProductoSeccionadoCod.getText().toString().trim()+"'");
6.if(productosDetalle.size()>0){//verfica si el produto esta activo
7. ArrayList<EmpresasProducto> empresaProduto=gEmpresasProductos.getEmpresasProductos(" WHERE empresasempresaproducto ='"+empleado.getEmpresas()+"' AND  producto
8. if(empresaProduto.size()>0){//verifica si el producto tiene pvp para poder agregar
9.empresaArray=gEmpresas.getEmpresas(" WHERE codigoempresa='"+empresaProduto.get(0).getEmpresasempresaproducto()+"'");
10.if(empresaArray.size()>0){
String codi_empresa=empresaArray.get(0).getCodigoempresa(); String nombre_empresa=empresaArray.get(0).getNombreempresa();
String cod_producto=lblProductoSeccionadoCod.getText().toString().trim();String cod_venta_producto=productosDetalle.get(0).getCodigoventaproducto();
String descri_producto=productosDetalle.get(0).getDescripcionproducto();double precio_producto=empresaProduto.get(0).getPrecioventaproducto();
11.extrasFacturar=gExtrasFacturar.getExtrasFacturar(" WHERE "+Esquema.TBLExtrasFacturar.COLUMN_NAME_PADREEXTRA+"='"+lblProductoSeccionadoCod.getText().to
//extrasFacturar verifica si tiene extras para agregar a la orden con el producto padre
12. if(extrasFacturar.size()>0){//
String codigoPadre="";  String codigoPadreOrpdetalle=ctrl.getCod();
OrpDetalleTemporal orp_det_temp =new OrpDetalleTemporal();
orp_det_temp.setCodigoOrpDetalle(codigoPadreOrpdetalle);
13.gOrpDetalleTemp.insertar(orp_det_temp);
mostrarDialogoExtras(codigoPadre,codi_empresa,nombre_empresa,cod_producto,cod_venta_producto,descri_producto,precio_producto,codigoPadreOrpdetall
//ingresa la orden con extras incluido
14. }else{
OrpDetalleTemporal orp_det_temp =new OrpDetalleTemporal();
orp_det_temp.setCodigoOrpDetalle(ctrl.getCod());
gOrpDetalleTemp.insertar(orp_det_temp);
//agrega el item q no tiene extras
Toast.makeText(getApplicationContext(),"Item Agregado a la order ",Toast.LENGTH_LONG).show();
}
15.}else{Toast.makeText(getApplicationContext(),"No encuentra la empresa",Toast.LENGTH_LONG).show();}
16. }else{Toast.makeText(getApplicationContext(),"El producto no pertenese a laempresa/no tiene precio ",Toast.LENGTH_LONG).show(); }
17. }else{
Toast.makeText(getApplicationContext(),"No existe Producto ",Toast.LENGTH_LONG).show();
18. }}else{}
}
19. }
```

Ilustración 110.Código para validar las entradas de Agregar Productos por catálogo a la Orden de Pedido. **Fuente:** El autor.

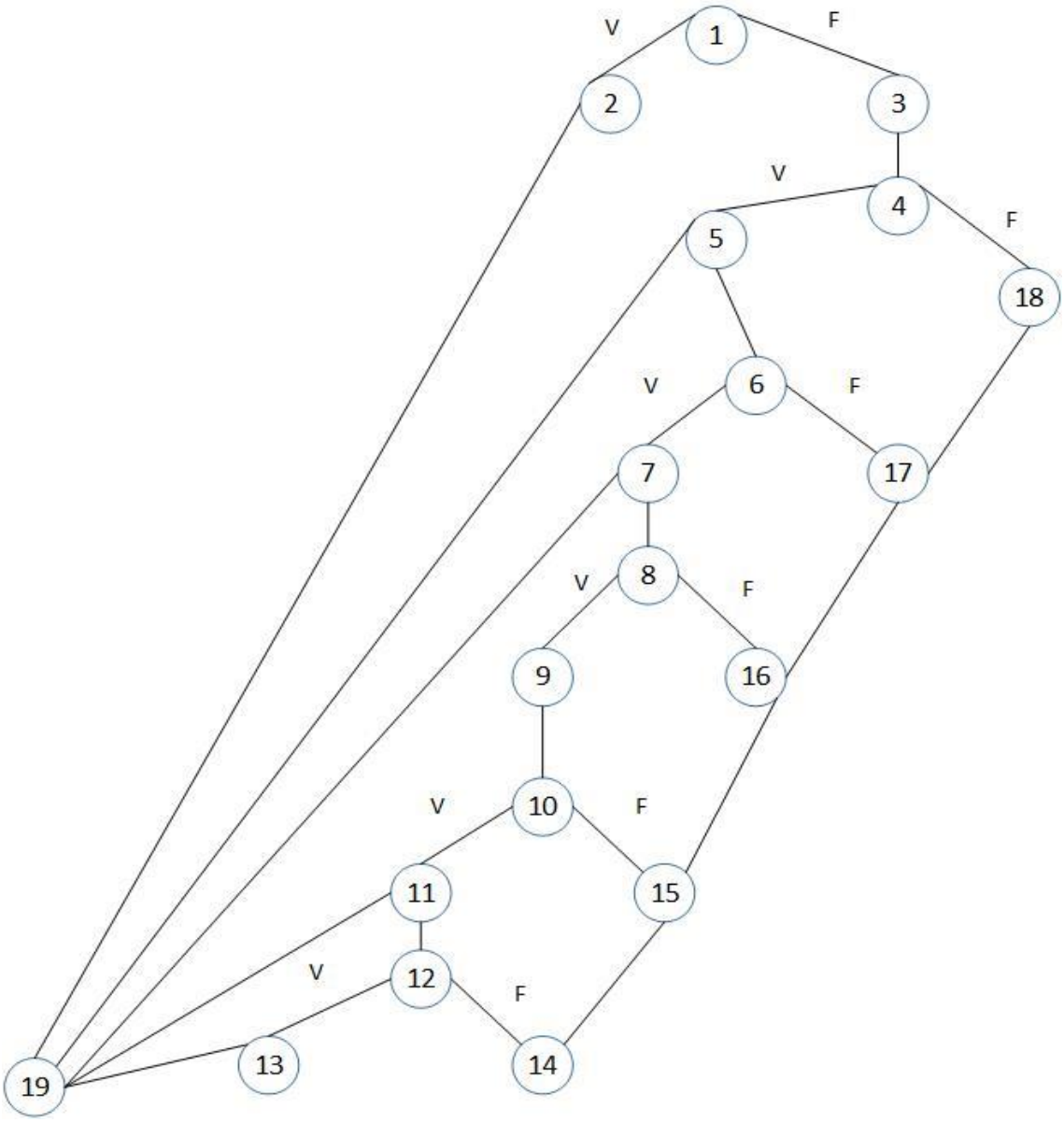


Ilustración 111. Grafo del camino principal para Agregar Productos al Pedido.
Fuente: El autor.

PRUEBA DE CAJA BLANCA-006: Agregar Productos por catálogo a la orden de pedido		
Fórmula: **V(G) = P + 1 =6 + 1 = 7**	***Número de caminos***	***Ruta de caminos independientes***
Donde: **V(G):** Complejidad ciclomática **P:**Nodos Predicados =6	**1**	1 – 12 – 19
	2	1 – 3 – 4 – 5 – 19
	3	1 – 3 – 4 – 5 – 6– 7 – 19
	4	1 – 3 – 4 – 5 – 6– 7 – 8– 9 – 10– 11 – 19
	5	1 – 3 – 4 – 5 – 6– 7 – 8– 9 – 10– 11 – 12 – 13– 19
	6	1 – 3 – 4 – 18 – 17– 16 – 15– 14 – 12 – 11– 19
	7	1– 3 – 4 – 18 – 17– 16 – 15– 14 – 12 – 13– 19

Tabla 44. Prueba de Caja Blanca-006: Número de caminos independientes. **Fuente:** El autor.

d) Prueba de Caja Blanca 007: Revisar el estado de la Orden de Pedido

```
private String estadoOrdenPedido(String estado,TextView color,String codigoOrdenCabecera){
    ArrayList<OrpCabecera> estadoOrdenCabecera=gOrpCabecera.getOrpCabecera("" +
                " WHERE "+Esquema.TBLOrpCabecera.COLUMN_NAME_CODIGO+"='"+codigoOrdenCabecera+"'  ");//consulta para ver el estado de pedido esta pendeinte o enviado
    String res="";
1.  if(estadoOrdenCabecera.size()>0){
2.    if(estadoOrdenCabecera.get(0).getEstadoOrpCabecera().equalsIgnoreCase("0")){//si es igual a cero pendiente
3.        estado="PENDIENTE";
          color.setBackgroundColor(Color.parseColor("#F23118"));
4.    }else{
          estado="ENVIADO";///estado enviado
          color.setBackgroundColor(Color.parseColor("#0E911D"));
      }
    }
    return res;

5. }
```

Ilustración 112.Código para validar las entradas del estado de la Orden de Pedido. **Fuente:** El autor.

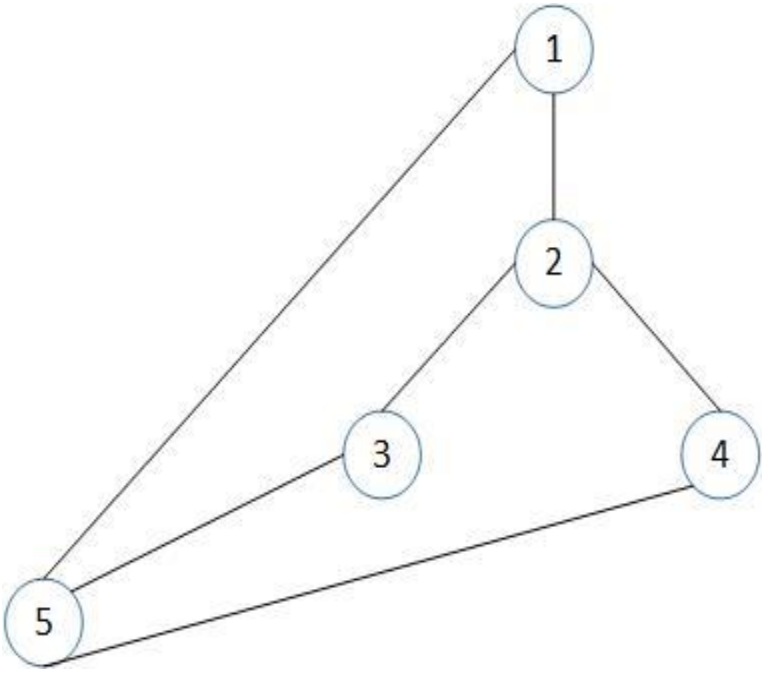


Ilustración 113. Grafo del camino principal para Listar Orden de Pedido. **Fuente:** El autor.

PRUEBA DE CAJA BLANCA-007: Listar Orden de Pedido		
Fórmula: **V(G) = P + 1 =2 + 1 = 3**	***Número de caminos***	***Ruta de caminos independientes***
Donde: **V(G):** Complejidad ciclomática **P:**Nodos Predicados =2	**1**	1 – 5
	2	1 – 2 – 3 – 5
	3	1 – 2– 4 – 5

Tabla 45. Prueba de Caja Blanca-007: Número de caminos independientes. **Fuente:** El autor.

5.3 Recopilación de Datos

5.3.1 Prueba de Usabilidad

El test de usabilidad aplicado a los siete vendedores generó las siguientes gráficas estadísticas.

PARTE I: CONTENIDO

- ***Pregunta 1***

  Le parece adecuada la selección de contenido presentado en el menú principal.

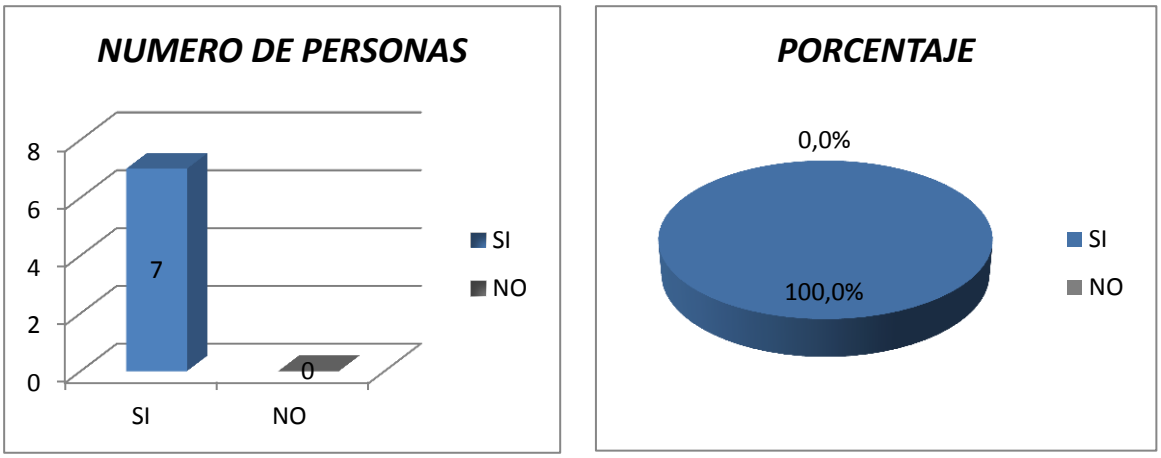


Ilustración 114. Resultado de la pregunta: ¿Le parece adecuada la selección de contenido presentado en el menú principal? **Fuente:** El autor.

- ***Pregunta 2***

  Al hacer click en los módulos de la aplicación ¿Halló en la información ofrecida lo que esperaba?

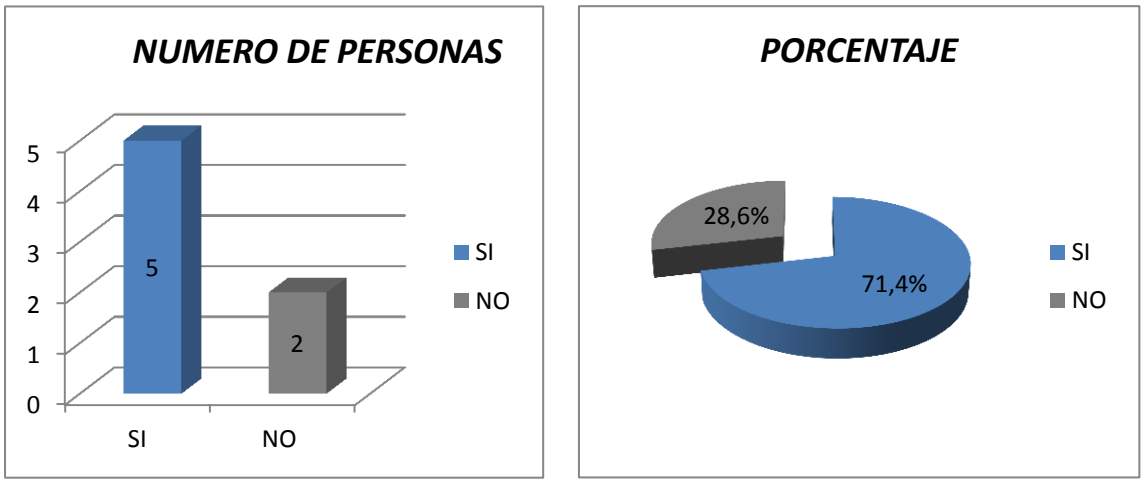


Ilustración 115.Resultado de la pregunta: Al hacer click en los módulos de la aplicación ¿Halló en la información ofrecida lo que esperaba? **Fuente:** El autor.

- ***Pregunta 3***

  Al hacer click en los módulos de la aplicación ¿Halló en la información ofrecida lo que esperaba?

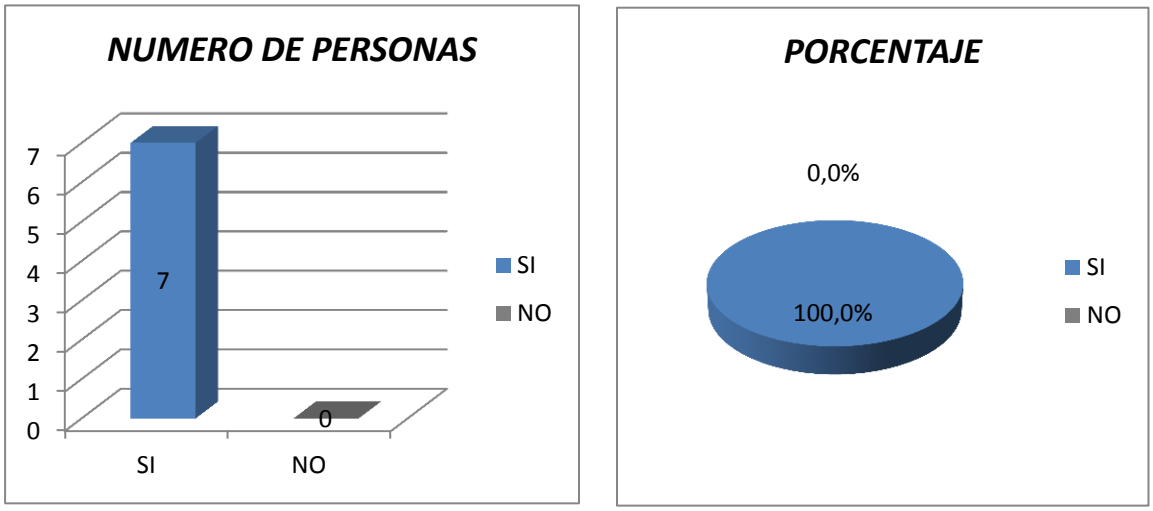


Ilustración 116. Resultado de la pregunta: Al hacer click en los módulos de la aplicación ¿Halló en la información ofrecida lo que esperaba? **Fuente:** El autor.

- ***Pregunta 4***

  Cree que los textos introductorios, cómo títulos, información de productos son claros y de fácil lectura.

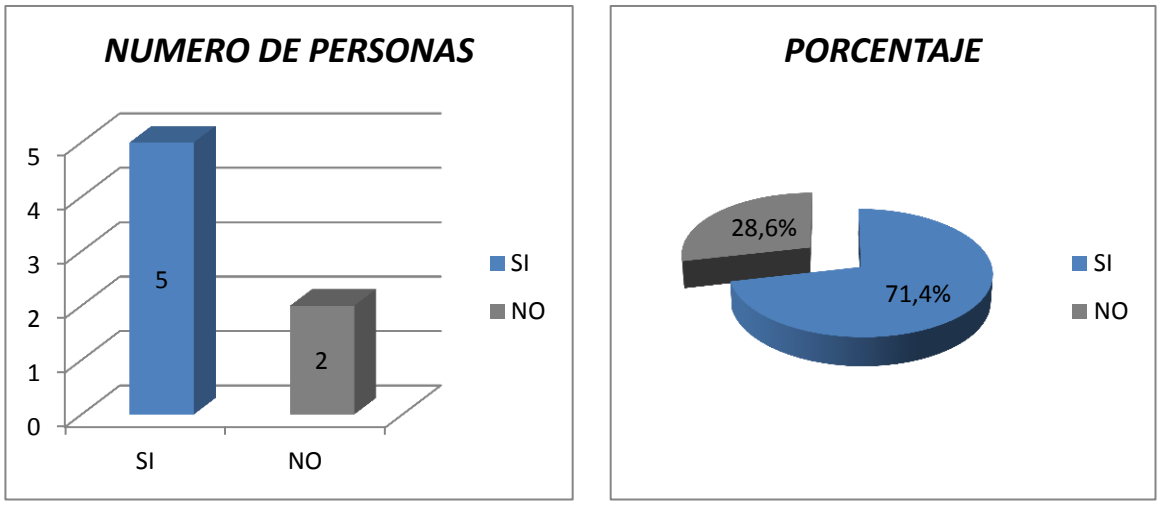


Ilustración 117. Resultado de la pregunta: Cree que los textos introductorios, cómo títulos, información de productos son claros y de fácil lectura**. Fuente:** El autor.

PARTE II: FACILIDAD DE APRENDIZAJE

- ***Pregunta 5***

  Las imágenes presentadas en la aplicación son reconocibles y describen claramente su función.

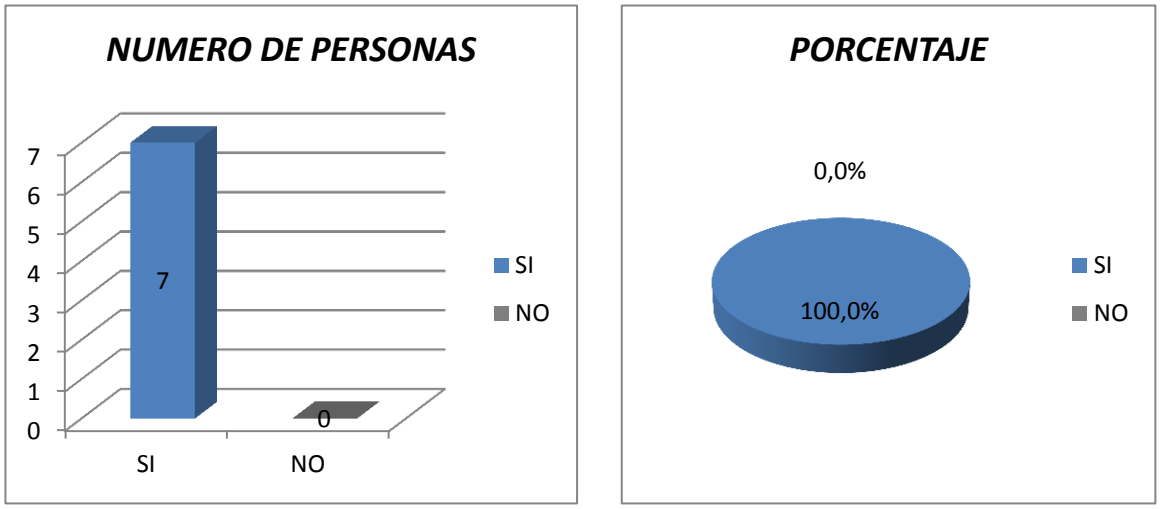


Ilustración 118. Resultado de la pregunta: Las imágenes presentadas en la aplicación son reconocibles y describen claramente su función. **Fuente:** El autor.

- ***Pregunta 6***

  Considera que las opciones secundarias de desplazamiento como: menú, opciones de subir, atrás, le ayudaron a usar la aplicación de mejor manera.

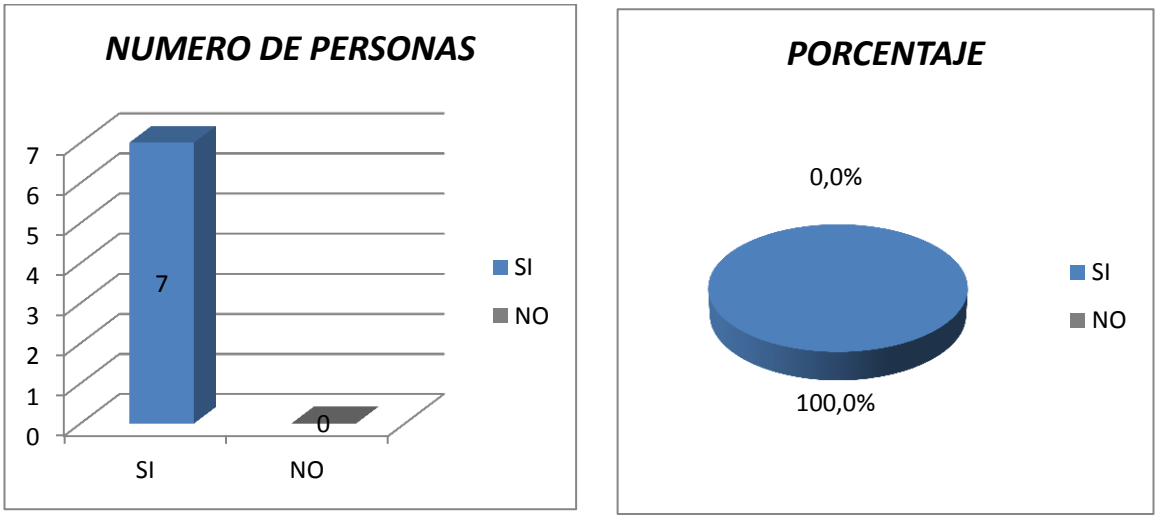


Ilustración 119. Resultado de la pregunta: Considera que las opciones secundarias de desplazamiento como: menú, opciones de subir, atrás, le ayudaron a usar la aplicación de mejor manera. **Fuente:** El autor.

- ***Pregunta 7***

  Los mensajes que presenta la aplicación son fáciles de entender

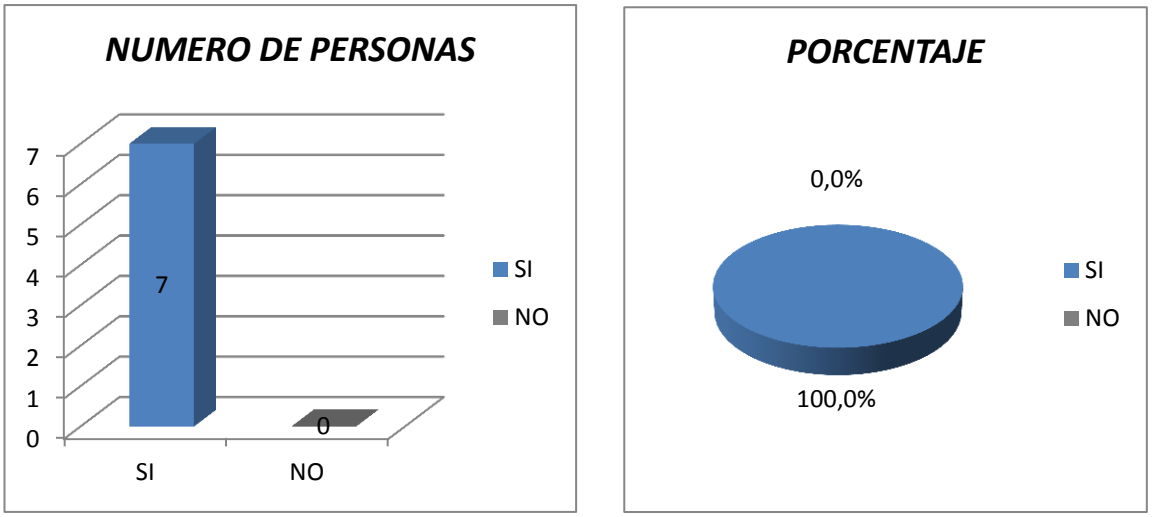


Ilustración 120. Resultado de la pregunta: Los mensajes que presenta la aplicación son fáciles de entender. **Fuente:** El autor.

Parte III: ACCESIBILIDAD

- ***Pregunta 8***

  Considera que el proceso de actualización de la cartera del cliente es satisfactorio.

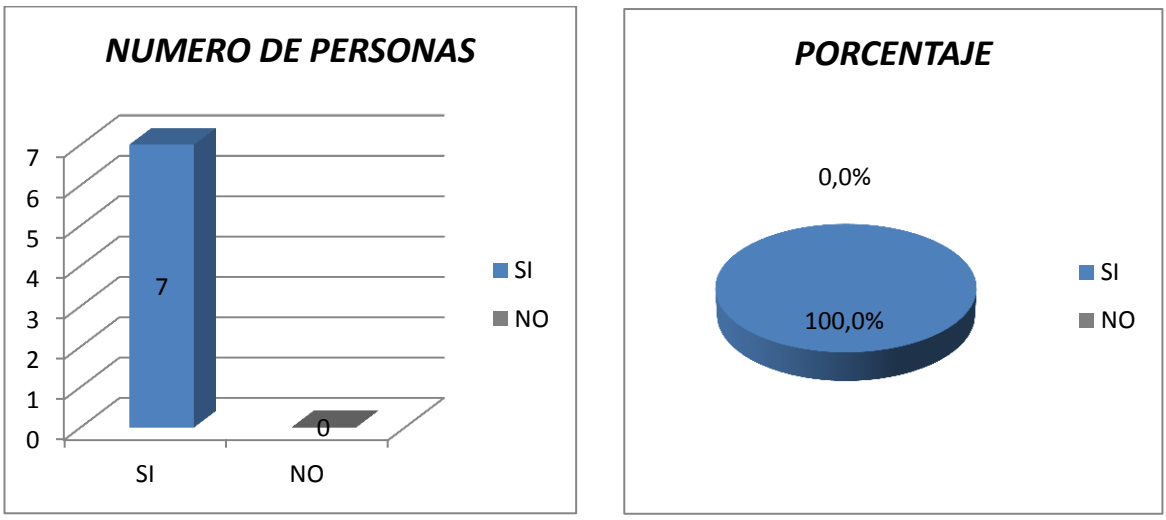
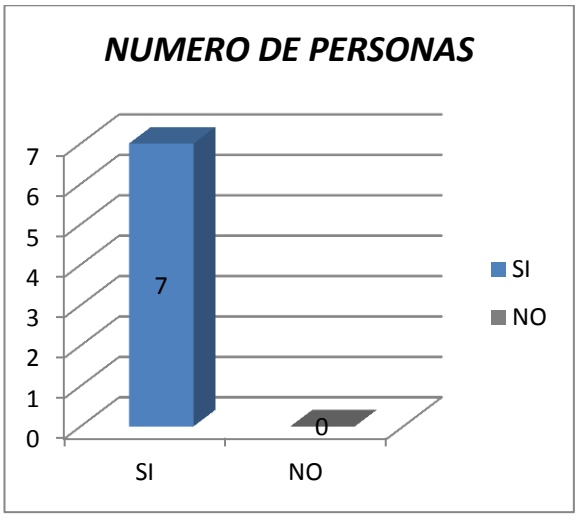


Ilustración 121. Resultado de la pregunta: Considera que el proceso de actualización de la cartera del cliente es satisfactorio. **Fuente:** El autor.

- ***Pregunta 9***

  El proceso de sincronización le garantiza a usted que los datos del móvil son iguales a los del servidor central.



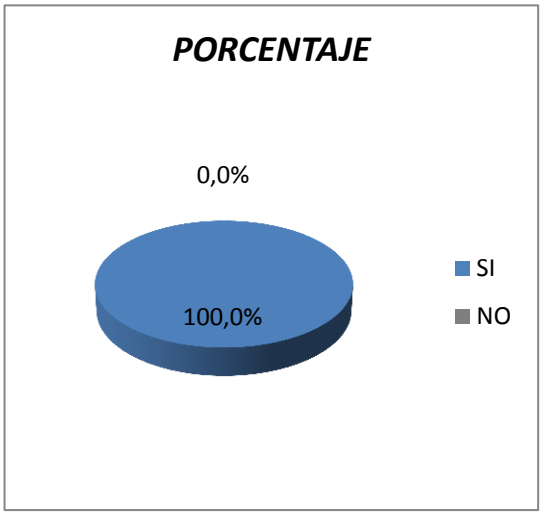
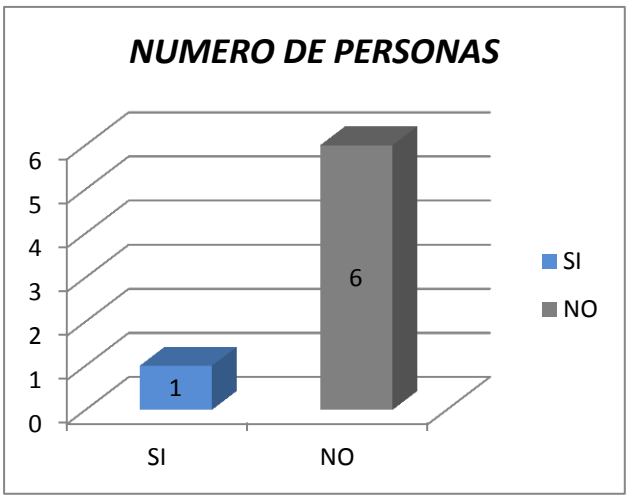


Ilustración 122. Resultado de la pregunta: El proceso de sincronización le garantiza a usted que los datos del móvil son iguales a los del servidor central. **Fuente:** El autor.

Parte IV: SATISFACCIÓN

- ***Pregunta 10***

  Existe algún elemento gráfico o texto que le ha causado confusión.



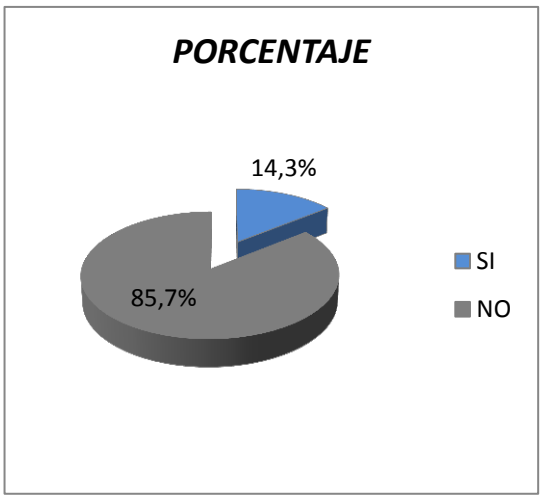


Ilustración 123. Resultado de la pregunta: Existe algún elemento gráfico o texto que le ha causado confusión. **Fuente:** El autor.

- ***Pregunta 11***

  La aplicación maneja estilos en cada una de sus pantallas.

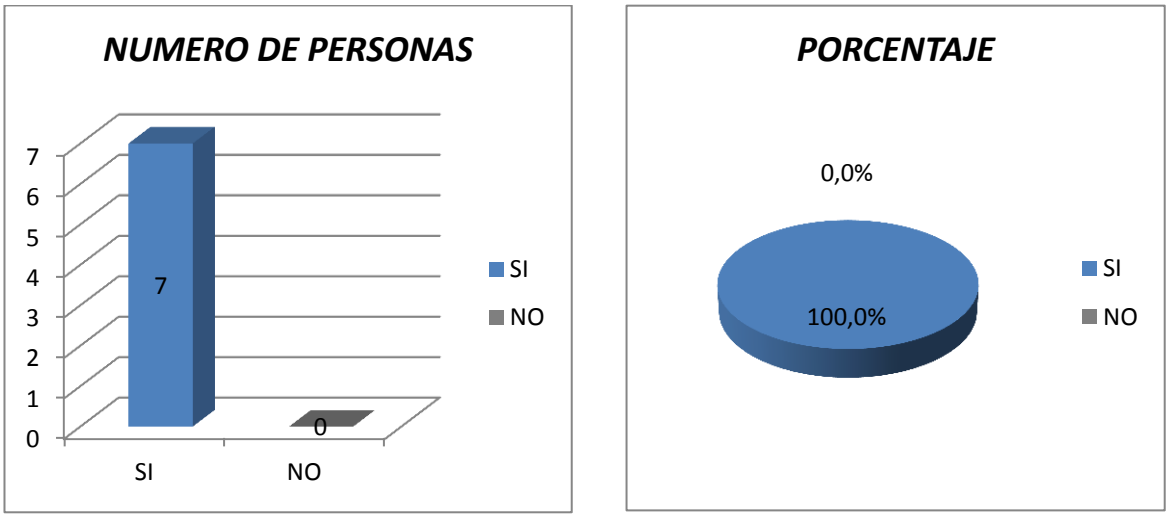


Ilustración 124. Resultado de la pregunta: La aplicación maneja estilos en cada una de sus pantallas**. Fuente:** El autor.

- ***Pregunta 12***

  La información de detalle de la orden de pedido es legible y correcta

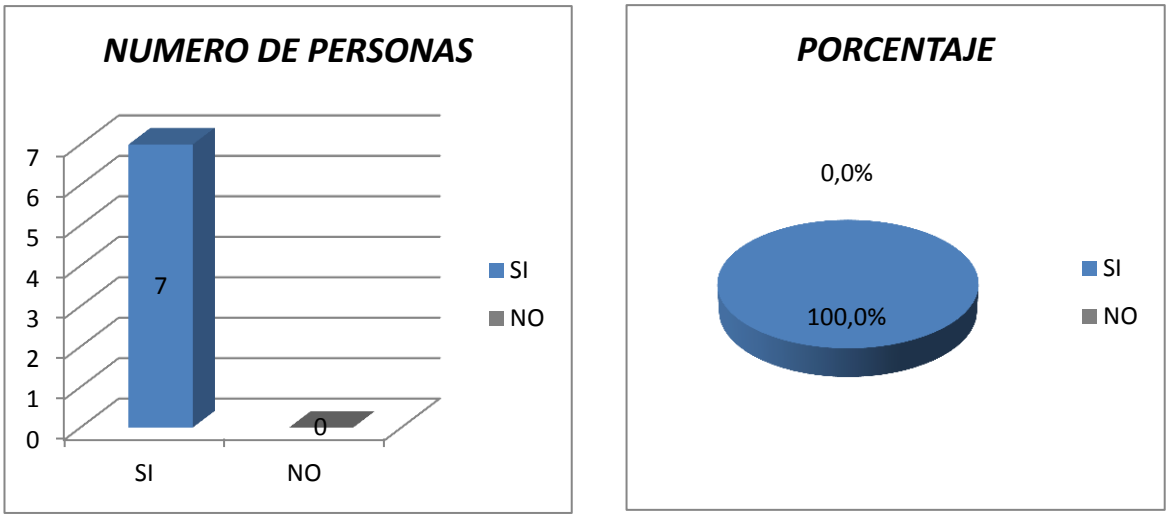
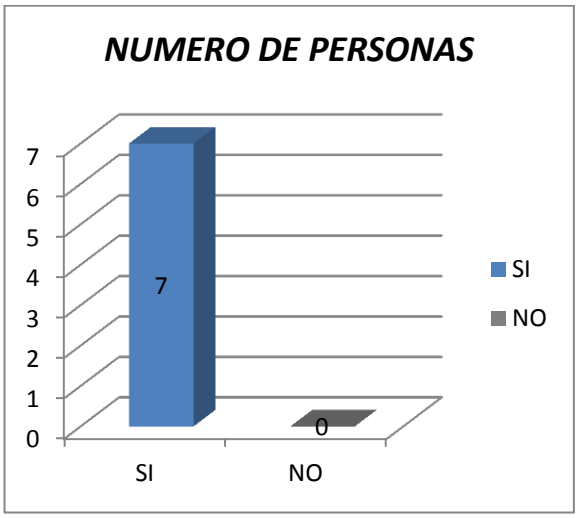


Ilustración 125. Resultado de la pregunta: La información de detalle de la orden de pedido es legible y correcta. **Fuente:** El autor.

Parte V: EFICIENCIA

- ***Pregunta 13***

  Cree usted que la aplicación móvil ha simplificado su trabajo operativo y ha agilizado el proceso de registro de pedidos.



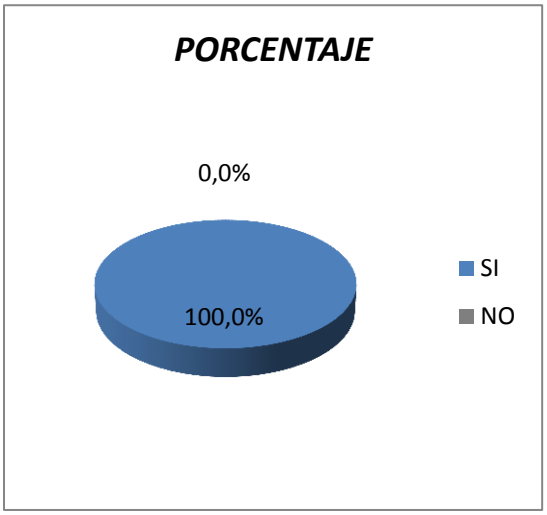
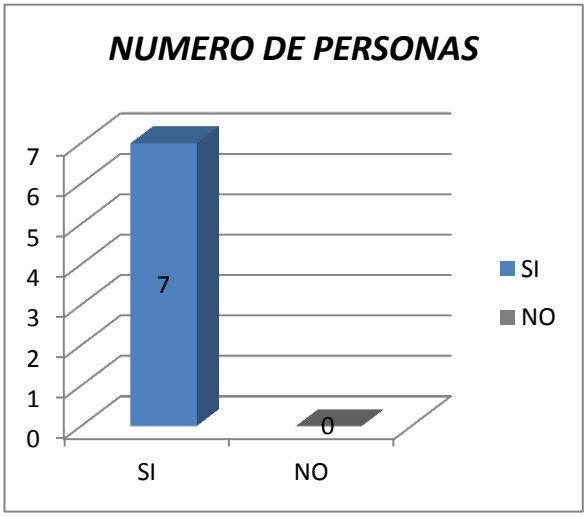


Ilustración 126. Resultado de la pregunta: La aplicación maneja estilos en cada una de sus pantallas. **Fuente:** El autor.

- ***Pregunta 14***

  En su opinión, la aplicación móvil facilita la comercialización de los productos y mejora la productividad de la empresa.



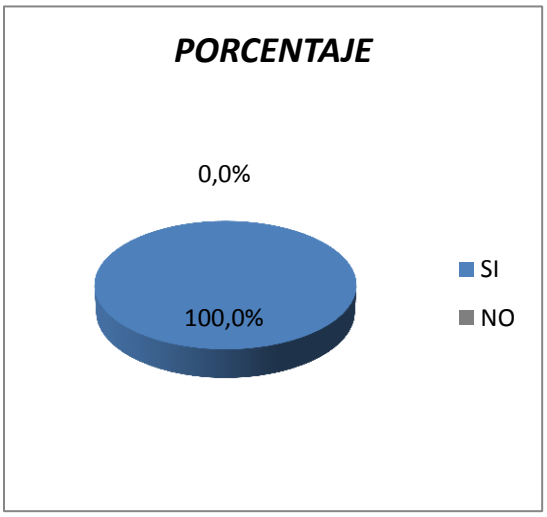


Ilustración 127. Resultado de la pregunta: En su opinión, la aplicación móvil facilita la comercialización de los productos y mejora la productividad de la empresa.
Fuente: El autor.

5.4 Análisis de resultados

5.4.1 Resultados de la Prueba de Usabilidad

Una vez obtenido las gráficas estadísticas es importante analizar los problemas encontrados para poder comenzar a plantear soluciones. Si bien la aplicación resulta muy intuitiva para el usuario, mediante la ejecución del test de usabilidad se pude detectar inconvenientes hasta ahora no visibles como:

- **Contenido:**

  Pregunta # 1: El 100% de encuestados respondieron que la aplicación está equilibrada y que el contenido presentado en el menú principal es adecuado.

  Pregunta # 2: El 71,4 % indicaron que siempre encontraron la información y que se facilitó el proceso de búsqueda, mientras que el 28,6% respondieron *NO* al no encontrar lo que esperaban al acceder al módulo cliente y módulo pedido. El vendedor uno indicó que al momento de realizar la búsqueda del cliente *" Zulagro ",* este no apreció ya que el cliente se encuentra en la base de datos como *" Sulagro S.A".* El vendedor dos indicó que la aplicación debería mostrar el perfil de autorización del pedido (gerente de ventas, gerente general, facturado).

  Pregunta # 3: El 100% de encuestados respondieron que si existen descripciones de ayuda dentro de cada módulo.

  Pregunta # 4: El 71, 4,1% respondieron afirmativamente a que los títulos, textos introductorios, información de productos son claros y de fácil lectura, mientras que el 28,6% respondieron *NO*. El primer vendedor indico que al realizar la búsqueda del producto*" Holladora"*, este no apareció dado que el producto se encuentra en la base de datos con el nombre de*" Hoyadora"*. El segundo vendedor sugirió ubicar una opción dónde exista divisiones de los productos tales como: repuestos, maquinaria, químicos, etc.

- **Facilidad de aprendizaje**: El 100% de encuestados indicaron afirmación acerca que las imágenes son claramente identificables, las opciones de menú, subir, atrás agilizan la interacción con la aplicación y los mensajes de advertencias, afirmación o error son fáciles de entender.

- **Accesibilidad:** Los siete vendedores indicaron que el proceso de sincronización es 100% exitoso al igual que la actualización de la cartera del cliente.

- **Satisfacción:** El 85,7% indicaron que los elementos gráficos no causan confusión, mientras el 14,3 % restante corresponde a la opinión de *NO* de un solo vendedor, esto se debe a que sugirió que los iconos de la barra de búsqueda sean más visibles. Por otro lado el 100% de encuestados indicaron que la aplicación maneja estilos en cada una de sus pantallas y la información de detalle de la orden es legible y correcta.

- **Eficiencia:** El 100% de vendedores indicaron la aplicación móvil agiliza su trabajo diario ya que permite colocar el pedido en el mismo instante, no necesitan llevar papeles extras, no se necesitan enviar correo de registro de pedido, se puede revisar la cartera del cliente actualizada, etc.

NOTA:

La aplicación WISE superó las expectativas del 85% de aprobación en cuanto a facilidad de aprendizaje, accesibilidad, satisfacción, eficiencia y contenido. Cabe recalcar que las mejoras propuestas por los vendedores serán implementadas para asegurar la experiencia del usuario.

5.4.2 Resultados de la Prueba de Funcionalidad

CASO DE PRUEBA – 001:	INGRESAR AL SISTEMA		
Autores:	Johanna Picón, Sebastián Zhinin		
Objetivo de la Prueba:	Asegurar el acceso a la aplicación WISE solo al personal autorizado.		
Actor:	Desarrolladores		
Condiciones de Éxito:	✓ Comprobar conexión con la ip del servidor central. ✓ Registrar el dispositivo móvil en el servidor central. ✓ El sistema no permitirá dejar campos en blanco. ✓ La aplicación no permitirá acceso a usuarios no registrados. ✓ Verificar el número de intentos que puede realizar el usuario para ingresar al sistema. ✓ Verificar la emisión de un mensaje al realizar la primera sincronización con el servidor.		
Puntos de Observación:	✓ Emitir un mensaje probando conectividad con el servidor central. ✓ Si el dispositivo móvil no ha sido registrado la aplicación no le permite acceso. ✓ Si el usuario deja campos en blanco, la aplicación emitirá un mensaje de error. ✓ Si el usuario o contraseña no coincide con el servidor central la aplicación no permite autenticarse. ✓ Verificar el bloqueo del usuario al ingresar tres intentos fallidos. ✓ Emitir un mensaje al realizar la primera sincronización para descargar los datos del servidor.		
Resultados Obtenidos:	**1.** La aplicación emite un mensaje indicando conexión con el servidor. **2.** Si la tablet no está previamente registrada la aplicación no permite el acceso. **3.** Los campos en blanco están validados. **4.** Al ingresar un usuario y contraseña incorrecta, la aplicación no permite el acceso. **5.** La clave ha sido bloqueada luego realizara tres intentos inválidos. **6.** La aplicación emite un mensaje de aviso al realizar la primera sincronización.	**¿Cumplido?** ☑ ☑ ☑ ☑ ☑ ☑	**Observaciones**

Tabla 46. Resultado Caso de Prueba- 001: Ingresar al Sistema. **Fuente:** El autor.

CASO DE PRUEBA – 002:	SINCRONIZAR DATOS DEL SERVIDOR CENTRAL		
Autores:	Johanna Picón, Sebastián Zhinin		
Objetivo de la Prueba:	Realizar la primera sincronización con el servidor para descargar los datos necesarios.		
Actor :	Desarrolladores		
Condiciones de Éxito:	✓ Verificar la emisión de un mensaje informativo indicando el inicio de descarga de información. ✓ Verificar la emisión de un mensaje de sincronización exitosa.		
Puntos de Observación:	✓ La aplicación emite un mensaje de sincronización exitosa. ✓ Los registros de eventos del servidor no presentan errores.		
Resultados Obtenidos:	**1.** La aplicación emite un mensaje indicando que la sincronización se estableció con éxito. **2.** El archivo de eventos del servidor no presenta errores.	**¿Cumplido?**	**Observaciones**
		☑ ☑	

Tabla 47. Resultado Caso de Prueba-002: Ingresar al sistema. **Fuente:** El autor.

CASO DE PRUEBA – 003:	LISTAR CLIENTE		
Autores:	Johanna Picón, Sebastián Zhinin		
Objetivo de la Prueba:	Listar la información de detalle del cliente		
Actor:	Desarrolladores		
Condiciones de Éxito:	✓ La aplicación lista todos los clientes registrados en el servidor central.		
Puntos de Observación:	✓ La información del cliente debe mostrar el siguiente detalle: nombres, apellidos, nombreComercial, teléfono, email.		
Resultados Obtenidos:	**1.** La aplicación emite una lista con toda la información del cliente.	**¿Cumplido?**	**Observaciones**
		☑	

Tabla 48. Resultado Caso de Prueba-003: Listar cliente. **Fuente:** El autor.

CASO DE PRUEBA – 004:	REGISTRAR ORDEN DE PEDIDO		
Autores:	Johanna Picón, Sebastián Zhinin		
Objetivo de la Prueba:	Permitir al vendedor registrar una orden de pedido exitosamente.		
Actor :	Desarrolladores		
Condiciones de Éxito:	✓ Verificar que el vendedor seleccione un cliente antes de intentar registrar la orden de pedido. ✓ Verificar que la orden contenga almenos un producto o item. ✓ El vendedor solo podrá seleccionar un item de cada grupo del combo. ✓ El vendedor está obligado a ingresar la observación de la orden de pedido. ✓ No se podrá dejar cero en el valor de la cantidad de ítems al momento de registrar una orden. ✓ Emitir un mensaje al grabar la orden de pedido.		
Puntos de Observación:	✓ Emitir un mensaje al no seleccionar previamente un cliente para registrar la orden. ✓ Emitir un mensaje indicando que la orden no contiene ítems. ✓ Emitir un mensaje de error al intentar seleccionar 2 o más ítems del grupo del combo. ✓ Emitir un mensaje al dejar en blanco el campo de observación de la orden. ✓ Emitir un mensaje cuando la cantidad de ítems de la orden sea cero. ✓ Emitir un mensaje de pedido guardado exitosamente.		
Resultados Obtenidos:		**¿Cumplido?**	**Observaciones**
	1. La aplicación emite un mensaje al no seleccionar un cliente antes de registrar la orden. **2.** Si la orden de pedido no contiene ítems se emite un mensaje de error. **3.** Al intentar escoger dos ítems de un mismo grupo del combo la aplicación emite un mensaje de error. **4.** Se emite un mensaje de error cuando no se ha ingresado la observación de la orden. **5.** Se emite un mensaje al dejar en cero el valor de la cantidad de ítems de la orden de pedido. **6.** La aplicación emite un mensaje de pedido grabado exitosamente.	☑ ☑ ☑ ☑ ☑ ☑	

Tabla 49. Resultado Caso de prueba-004: Registrar Orden de Pedido. **Fuente:** El autor.

CASO DE PRUEBA – 005:	LISTAR PRODUCTOS		
Autores:	Johanna Picón, Sebastián Zhinin		
Objetivo de la Prueba:	Permitir listar productos de acuerdo a una categorización.		
Actor:	Desarrolladores		
Condiciones de Éxito:	✓ Listar productos por categorías, líneas o marcas. ✓ Identificar en qué categorización está el producto. ✓ Verificar la descripción de detalle del producto.		
Puntos de Observación:	✓ Se emite un mensaje indicando la categorización a la que pertenece el producto. ✓ El detalle del producto presenta la siguiente información: nombre, descuento, PVP, precioCliente, codigoProducto.		
Resultados Obtenidos:	**1.** La aplicación emite una lista con los productos existentes en esa categorización. **2.** La aplicación muestra descripciones indicando la categorización a la que pertenece el producto. **3.** Cada producto presenta el siguiente detalle: nombre, descuento, PVP, precioCliente, codigoProducto.	**¿Cumplido?**	**Observaciones**

Tabla 50. Resultado Caso de prueba-005: Listar Productos. **Fuente:** El autor.

CASO DE PRUEBA – 006:	AGREGAR PRODUCTOS POR CATALOGO A LA ORDEN DE PEDIDO		
Autores:	Johanna Picón, Sebastián Zhinin		
Objetivo de la Prueba:	Permitir agregar productos por catálogo directamente al pedido.		
Actor:	Desarrolladores		
Condiciones de Éxito:	✓ Comprobar la emisión de un mensaje al agregar productos correctamente al pedido.		
Puntos de Observación:	✓ Se emite un mensaje cuando la orden de pedido no contiene productos. ✓ Verificar el descuento por líneas de cada producto. ✓ Verificar que el producto tenga PVP para poder agregarlo a la orden. ✓ Verificar si el producto tiene extras o combo.		
Resultados Obtenidos:	**1.** La aplicación emite un mensaje de confirmación al momento de agregar items a la orden. **2.** Si la orden de pedido no contiene almenos un item, se emite un mensaje de error. **3.** El descuento de cada producto se aplica por línea. **4.** Todo producto tiene un precio de venta al público (PVP). **5.** Si el producto tiene extra, la aplicación muestra el cómo asociado.	**¿Cumplido?**	**Observaciones**

Tabla 51. Caso de prueba-006: Agregar productos pro catálogo a la orden de pedido.

CASO DE PRUEBA – 007:	LISTAR ORDEN DE PEDIDO		
Autores:	Johanna Picón, Sebastián Zhinin		
Objetivo de la Prueba:	Comprobar que el sistema liste las ordenes de pedido y permita visualizar el estado de la orden.		
Actor:	Desarrolladores		
Condiciones de Éxito:	✓ Listar todos los pedidos registrados aplicando un filtro por fecha o por cliente. ✓ El sistema permita visualizar el cambio de estado del pedido (pendiente o enviado).		
Puntos de Observación:	✓ Se emite un mensaje indicando que no existen pedidos en ese rango de fecha. ✓ Se emite un mensaje al no tener órdenes asociadas a ese cliente.		
Resultados Obtenidos:	**1.** La aplicación lista todos los pedidos filtrados por fecha o por cliente. **2.** Si no existen pedidos en un rango de fecha se emite un mensaje informativo. **3.** Si el cliente no tiene pedidos registrados se emite un mensaje indicándolo. **4.** La aplicación permite visualizar el cambio de estado del pedido de pendiente a enviado.	**¿Cumplido?**	**Observaciones**

Tabla 52. Caso de prueba-007: Listar orden de pedido. **Fuente:** El autor.

CASO DE PRUEBA – 008:	LISTAR CARTERA DEL CLIENTE		
Autores:	Johanna Picón, Sebastián Zhinin		
Objetivo de la Prueba:	Listar la cartera del cliente por un rango de fecha.		
Actor:	Desarrolladores		
Condiciones de Éxito:	✓ El sistema permite listar la cartera de un cliente activo.		
Puntos de Observación:	✓ Se emite un mensaje indicando que el cliente no tiene una cartera activa.		
Resultados Obtenidos:	**1.** La aplicación lista la cartera del cliente activo **2.** Si un cliente no tiene cartera, la aplicación emite un mensaje indicando.	**¿Cumplido?**	**Observaciones**

Tabla 53. Caso de prueba-008: Listar orden de pedido. **Fuente:** El autor.

CASO DE PRUEBA – 009: ACTUALIZAR CARTERA DEL CLIENTE			
Autores:	Johanna Picón, Sebastián Zhinin		
Objetivo de la Prueba:	Permitir actualizar el detalle de cartera de un cliente concreto.		
Actor:	Desarrolladores		
Condiciones de Éxito:	✓ El sistema permite actualizar la cartera de un cliente específico.		
Resultados Obtenidos:	✓ Se emite un mensaje indicando la actualización exitosa de la cartera del cliente.		
Resultados Obtenidos:	**1.** La aplicación móvil permite actualizar la cartera de un cliente específico y no todo el estado.	**¿Cumplido?**	**Observaciones**
		☑	

Tabla 54. Resultado Caso de prueba-009: Actualizar cartera del cliente. **Fuente:** El autor.

CASO DE PRUEBA – 010: REGISTRAR HOJA DE VISITA			
Autores:	Johanna Picón, Sebastián Zhinin		
Objetivo de la Prueba:	Registrar los cobros, clientes nuevos realizados por el vendedor.		
Actor:	Desarrolladores		
Condiciones de Éxito:	✓ El sistema registra la información de cobros, registro de visita y clientes nuevos que han sido ingresados al móvil.		
Resultados Obtenidos:	✓ Se emite un mensaje indicando que la información se almaceno correctamente en la base de datos móvil.		
Resultados Obtenidos:	**2.** La aplicación WISE permite registrar los cobros, clientes nuevos y el estado de visita realizado por el vendedor.	**¿Cumplido?**	**Observaciones**
		☑	

Tabla 55. Resultado Caso de prueba-009: Registrar Hoja de Visita. **Fuente:** El autor.

A continuación se presenta un reporte del número de registro de pedidos que ha ingresado cada vendedor a través del sistema móvil, dando un total de 284 pedidos ingresados satisfactoriamente, los mismos que fueron despachados.

ID Equipo	Vendedores	Total Ingresados
634121354	CRHISTIAN EDUARDO ARMAS MUGMAL	80
-1188425355	ALEJANDRO ENRIQUE CALLE ROCA	20
634121883	ROBERTO MARCO CHANG ANGULO	15
634389549	YURI SIMON FARIAS SOLEDISPA	15
-1888332345	JUAN PABLO LEON AVECILLAS	120
233800879	CARLOS SANTIAGO TORRES ROBLES	22
634389477	KLEVER VINICIO PERALTA FONSECA	12
	Total	**284**

Tabla 56. Reporte de pedidos ingresados desde el móvil. **Fuente:** El autor.

5.5 Manual de funcionalidad para el administrador

El siguiente manual permite al administrador crear el archivo APK, el cual se utilizará para instalar la aplicación en las tablets Android.

MANUAL DE ADMINISTRADOR

PASO 1: *Click derecho sobre el proyecto* → *Android Tools*

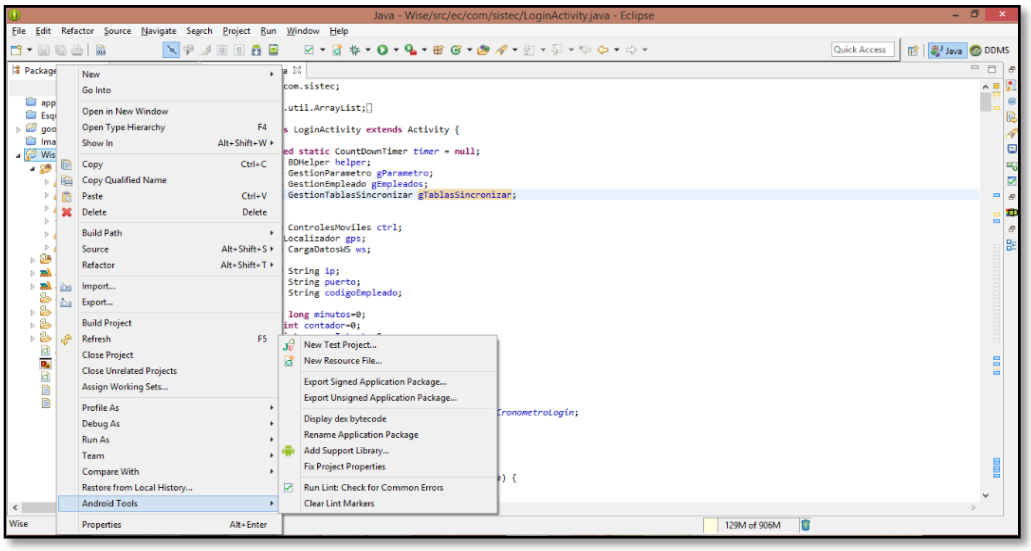


Ilustración 128. Explorador de paquetes de Eclipse. **Fuente:** El autor.

PASO 2: Seleccionar la opción *Export Signed Appliaction Package*

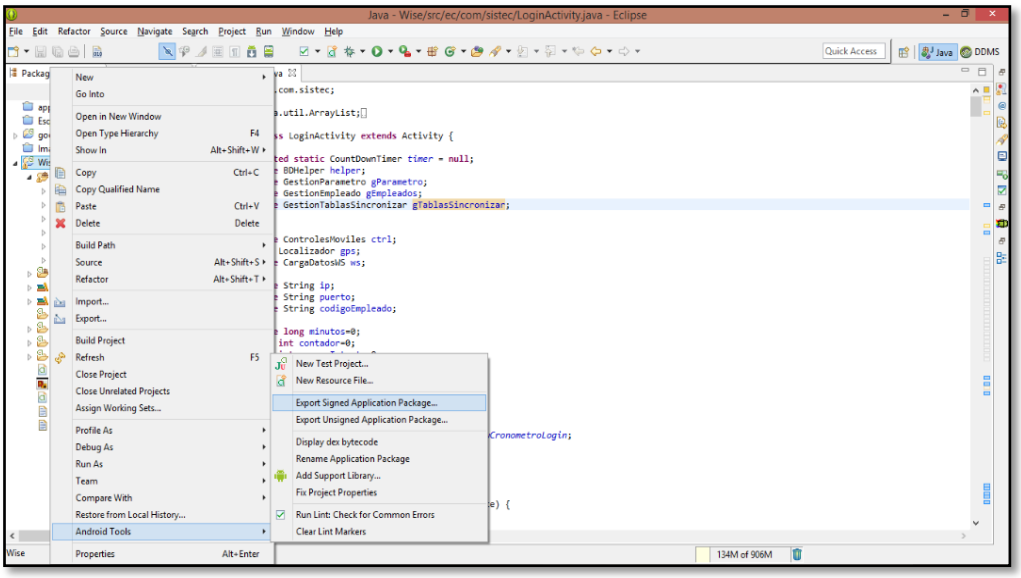


Ilustración 129. Exportación del proyecto Android. **Fuente:** El autor.

PASO 3: Buscar el proyecto que se desea exportar y dar click en *Next*

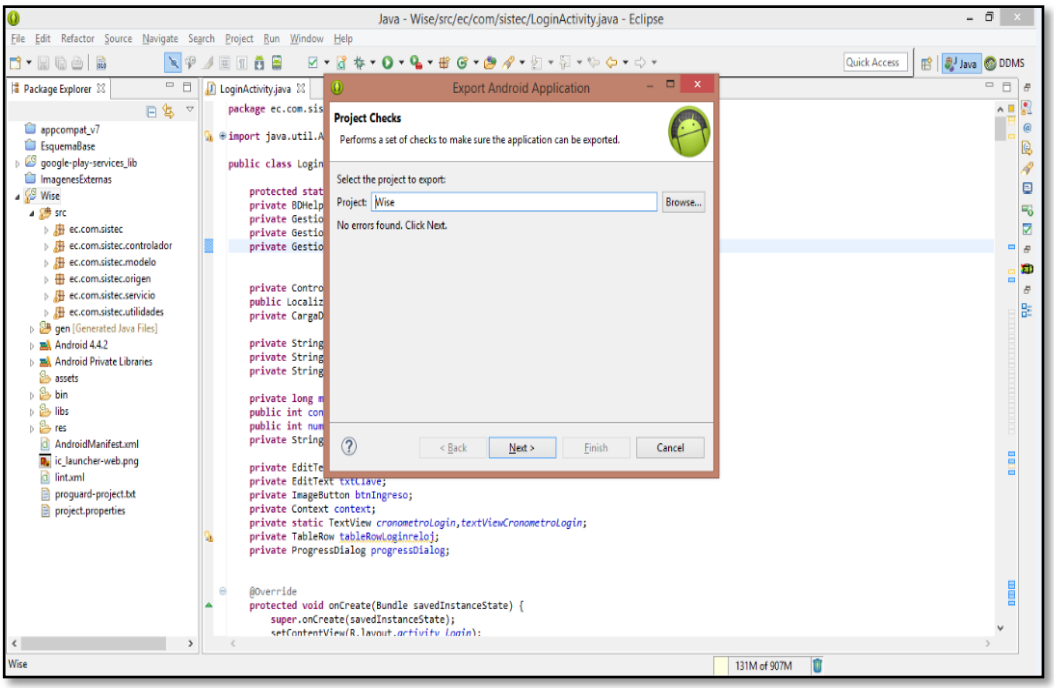


Ilustración 130. Selección del proyecto a exportar. **Fuente:** El autor.

PASO 4: Para publicar una aplicación Android necesitamos crear un APK firmado digitalmente. Aquí podemos optar por utilizar una clave existente seleccionando *Use Existing Keystore* o crear una nueva escogiendo *Create New Keystore.*

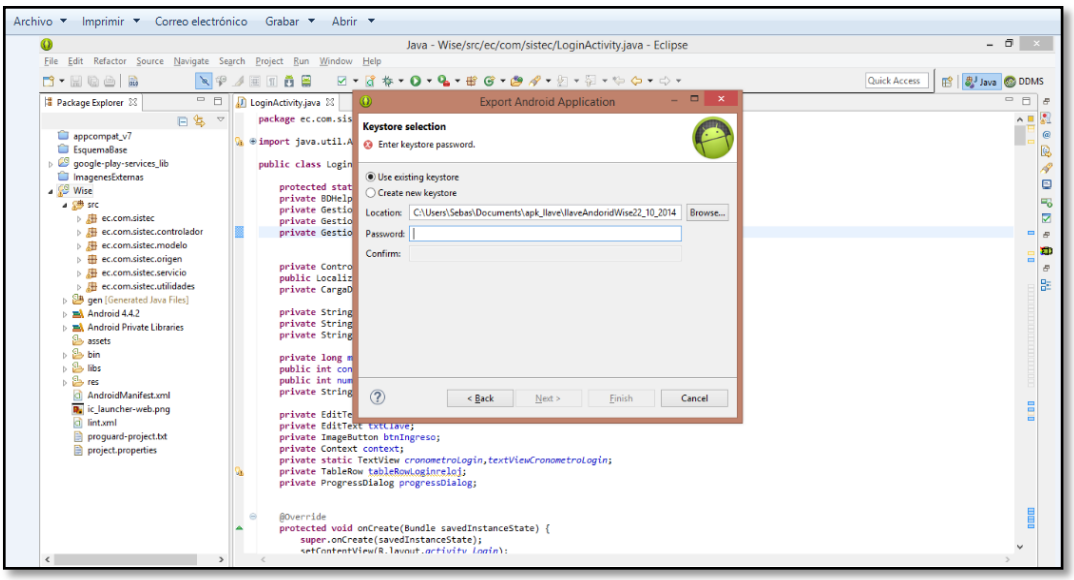


Ilustración 131. Selección de una clave existente. **Fuente:** El autor.

PASO 5: En el caso de haber escogido utilizar una clave existente lo único que haremos es escribir la contraseña y dar click en *Next.*

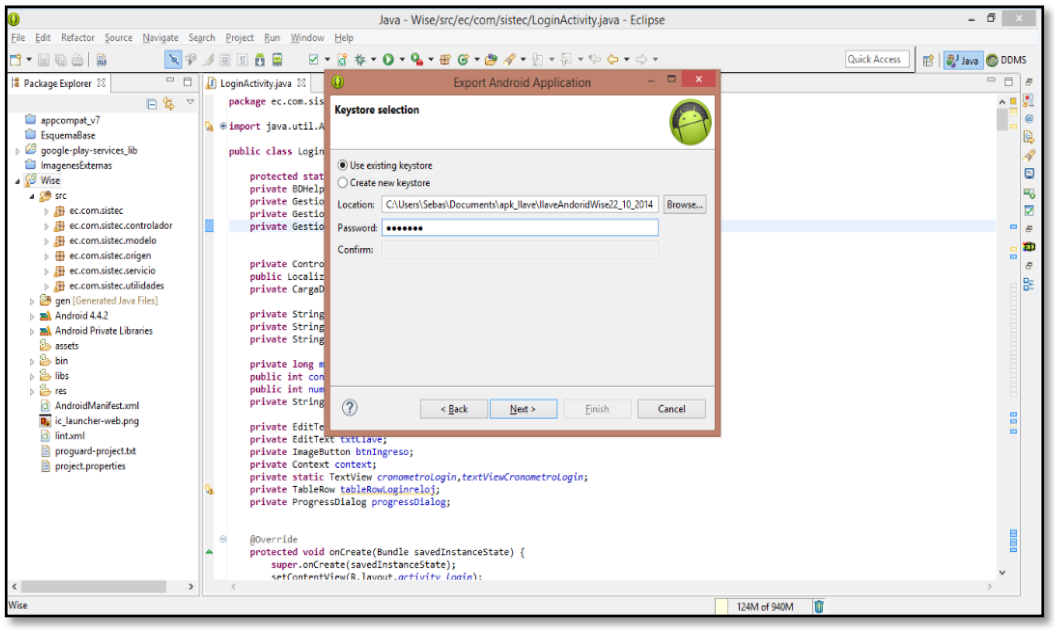


Ilustración 132. Selección del almacén de claves. **Fuente:** El autor.

PASO 6: En la siguiente pantalla debemos seleccionar la opción *Use Existing Key*, seleccionar el alias ya creado y digitar el password correspondiente.

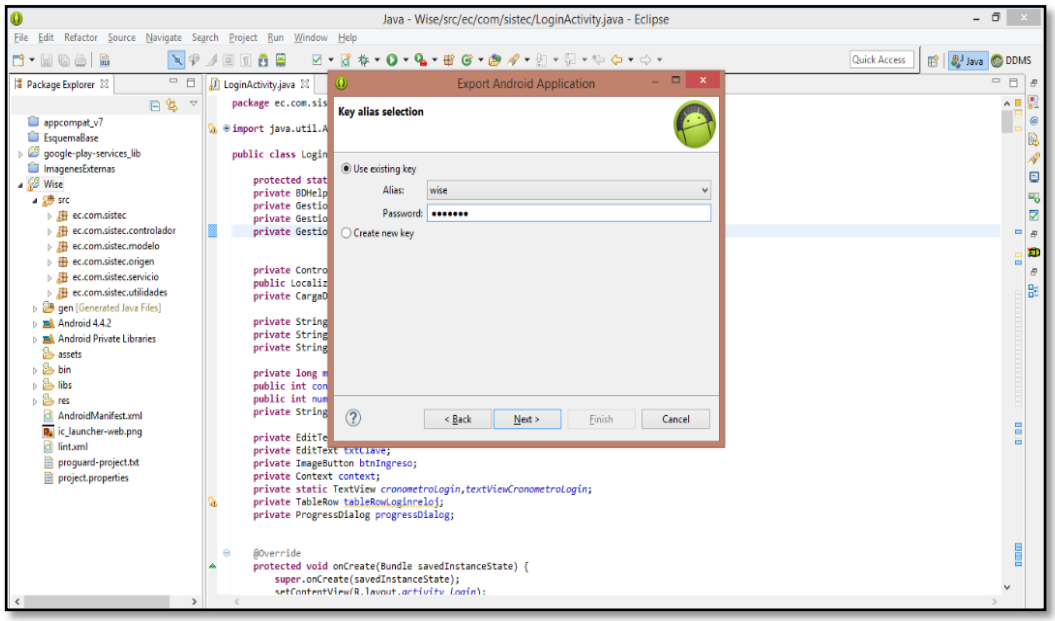


Ilustración 133. Selección del alias del certificado digital. **Fuente:** El autor.

PASO 7: La siguiente imagen describe la creación del certificado digital indicando la ubicación donde será almacenado, la fecha de expiración, etc. Aquí damos click en *Finish.*

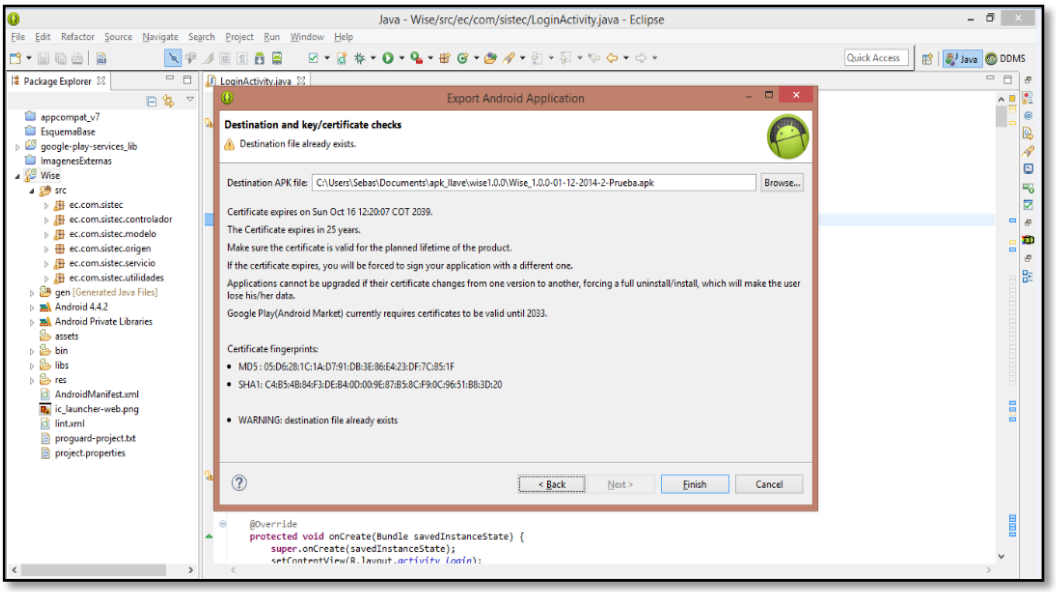


Ilustración 134. Creación del certificado digital en Android. **Fuente:** El autor.

PASO 8: Por ultimo podemos visualizar el archivo APK creado y listo para su instalación en un dispositivo Android.

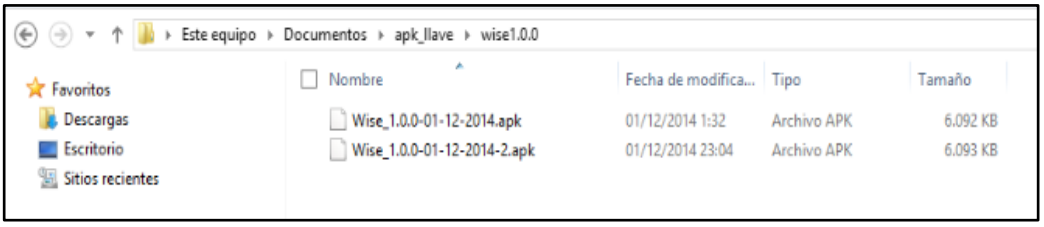


Ilustración 135. Archivo APK WISE. **Fuente:** El autor.

5.6 Manual de funcionalidad para el usuario

MANUAL DE USUARIO

PASO 1: Ejecutar el instalador *Wise_Instalador-30_11_2014-1.apk*

Ilustración 136. Descarga del instalador de la aplicación WISE. **Fuente:** El autor.

PASO 2: La siguiente pantalla indica las condiciones de privacidad de la aplicación WISE. A continuación damos click en *Siguiente*

Ilustración 137. Instalación de la aplicación WISE. **Fuente:** El autor.

PASO 3: También se detalla las condiciones de acceso al dispositivo como: ver el estado de la red, ver el estado de Wifi, acceso a internet ilimitado, etc. Aquí damos click en *Instalar*.

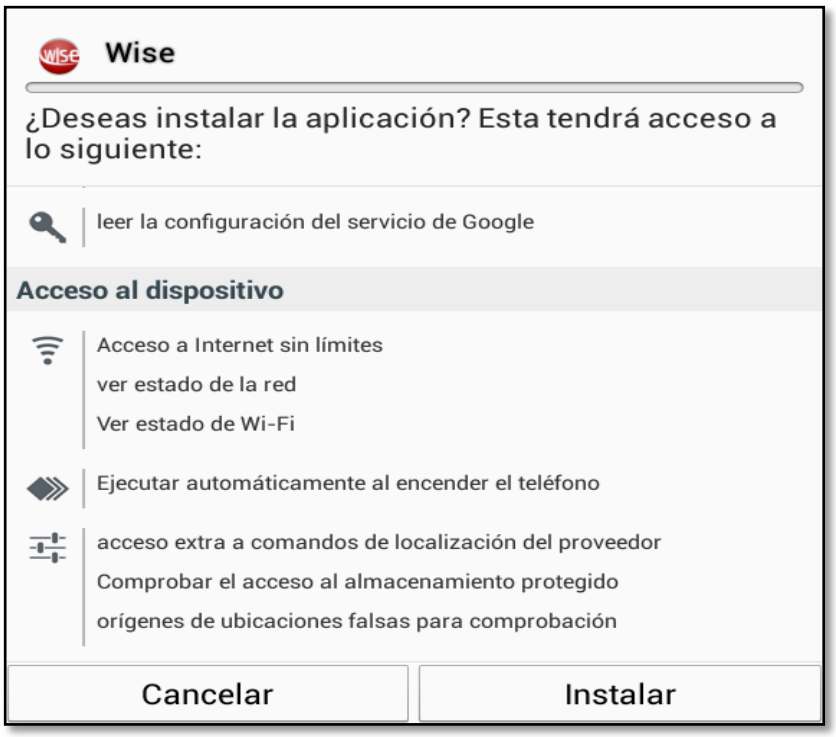


Ilustración 138. Instalación de las condiciones de acceso de WISE. **Fuente:** El autor

PASO 4: Una vez concluido el proceso de instalación se presentará la siguiente imagen.

Ilustración 139.Instalación completa de WISE. Fuente: El autor.

PASO 5: Procedemos a realizar la configuración inicial ingresando la dirección IP del proveedor y el número de puerto. Luego damos click en *Aceptar*.

Ilustración 140. Configuración de la dirección IP del proveedor. **Fuente:** El autor.

PASO 6: Una vez realizada la primera sincronización con el servidor central debemos esperar unos minutos hasta que todos los datos hayan sido descargados del servidor. Este paso solo se realiza una vez después se puede trabajar localmente.

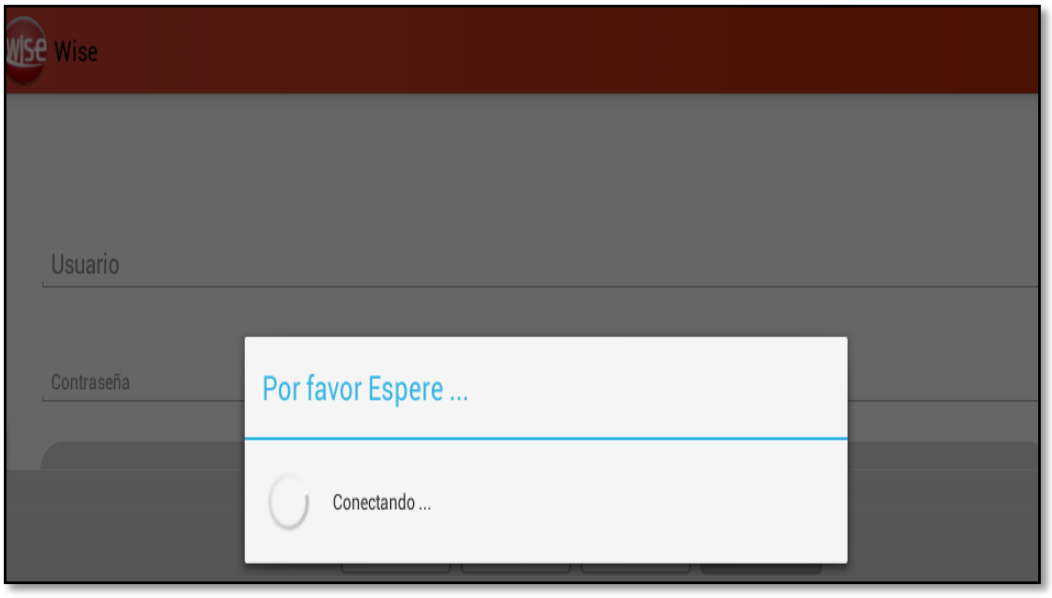


Ilustración 141. Mensaje de establecimiento de conexión con el servidor**. Fuente:** El autor.

PASO 7: Si la sincronización se realizó con éxito se visualizará la siguiente pantalla de logueo. Aquí el vendedor deberá ingresar su usuario y contraseña previamente registrado en el servidor central para poder autentificarse.

Ilustración 142. Interfaz de ingreso al sistema. Fuente: El autor.

PASO 9: Ingreso al menú principal de la aplicación WISE, allí se podrá visualizar los seis módulos que conforman el sistema y los elementos de la barra de acción.

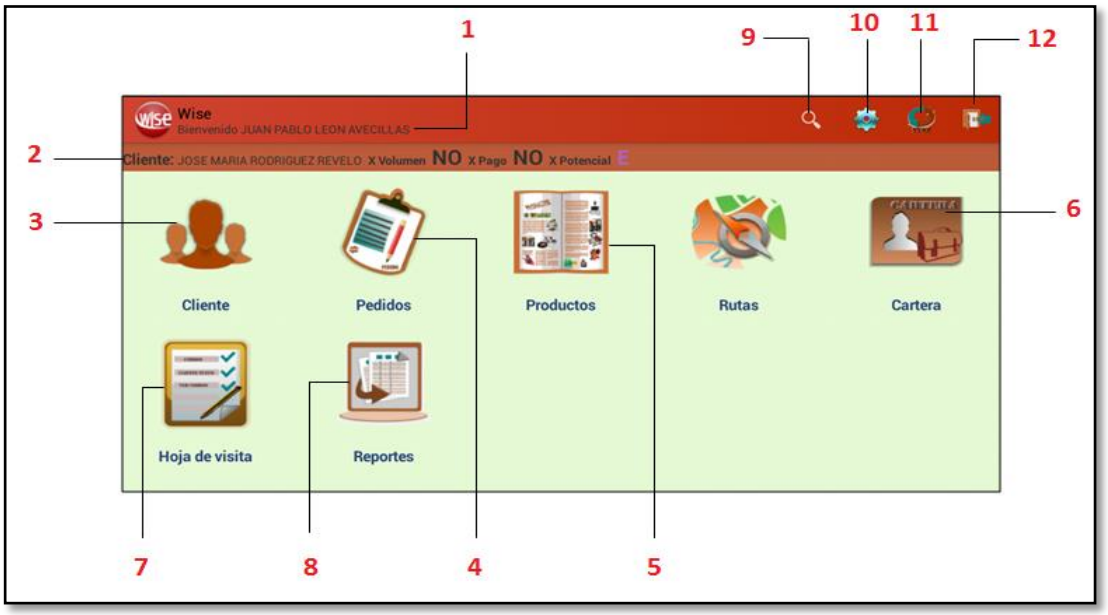


Ilustración 143. Interfaz del Menú Principal. **Fuente:** El autor.

1. **Nombre del vendedor**
2. **Nombre del cliente y calificación**
3. **Módulo Cliente**
4. **Módulo Pedidos**
5. **Módulo Productos**

6. **Módulo Cartera**
7. **Módulo Hoja de visita**
8. **Módulo Reportes**
9. **Buscador**
10. **Información de la dirección IP**
11. **Botón de sincronización**
12. **Salir**

- ***Módulo Cliente***

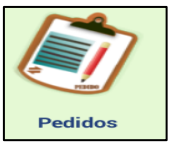
Incluye los datos de un determinado cliente como nombre, teléfono, email, etc.

1. **Datos del cliente:** nombres, nombre comercial, teléfono, email. :
2. **Botón Cartera:** Redirecciona al Módulo Cartera.
3. **Botón Ordenes de Pedido:** Redirecciona al Módulo Ordenes de Pedido.

Ilustración 144. Interfaz módulo cliente. **Fuente:** El autor.

- ***Módulo Pedido***

1. Diríjase a la opción de búsqueda del menú principal para seleccionar un cliente.

2. Acceder al *"Módulo Pedidos"* 

y seleccionar el submódulo *"Nueva Orden"*.

Ilustración 145. Submódulo Nueva Orden y Ver Ordenes. **Fuente:** El autor.

NOTA:

Si usted no ha seleccionado un cliente previamente, no podrá registrar una *"Nueva Orden"*.

3. La interfaz *"Nueva Orden"* contendrá el detalle de nombre del cliente, calificación del cliente, forma de pago, valor total y subtotal del pedido y la observación. Como podemos observar en la siguiente ilustración.

Ilustración 146. Registro de la orden de pedido. **Fuente:** El autor.

4. Procedemos a buscar un producto dando click en el botón *Buscar Producto.* Allí se listará las coincidencias encontradas.

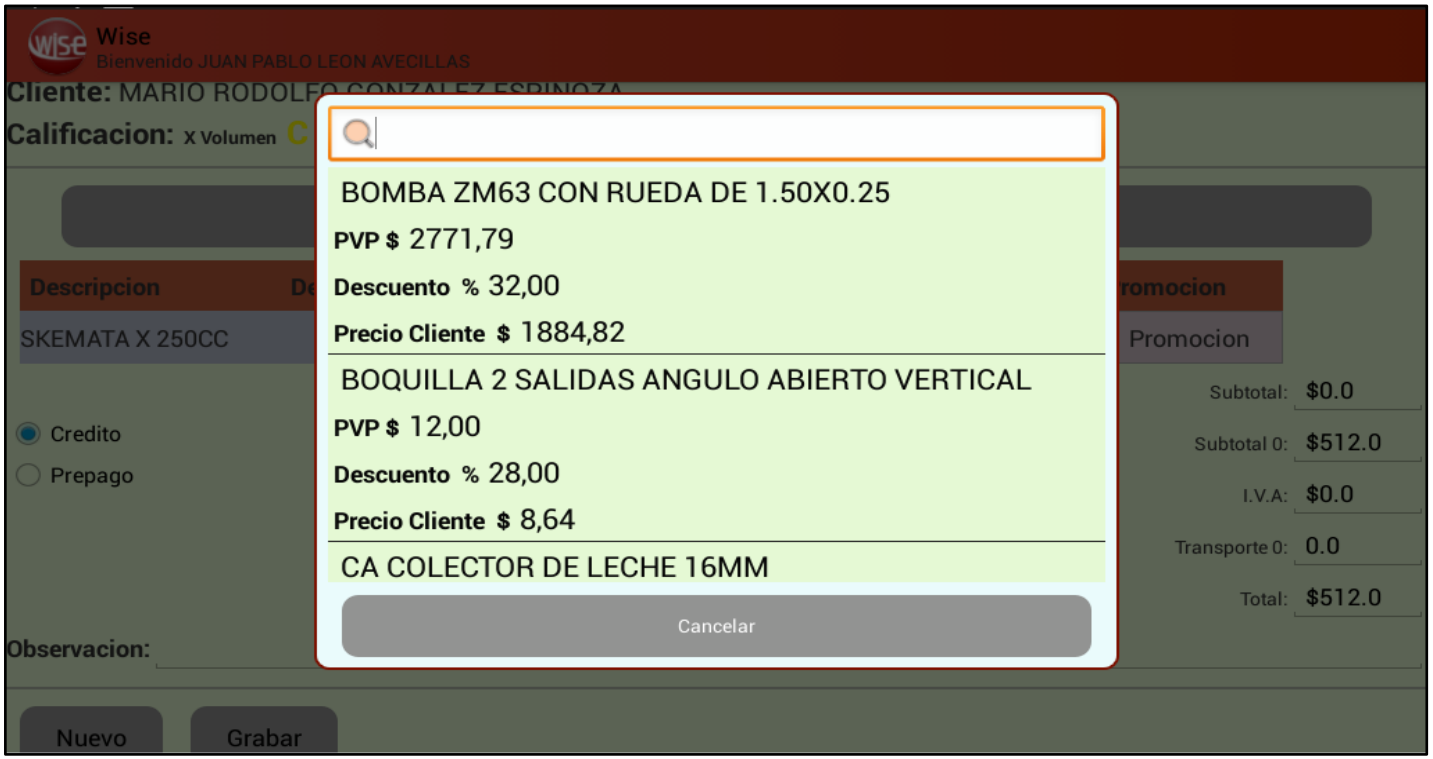


Ilustración 147.Búsqueda de productos para agregar a la orden. **Fuente:** El autor.

5) Si el ítem seleccionado tiene extras se deberá seleccionar un ítem de cada grupo.

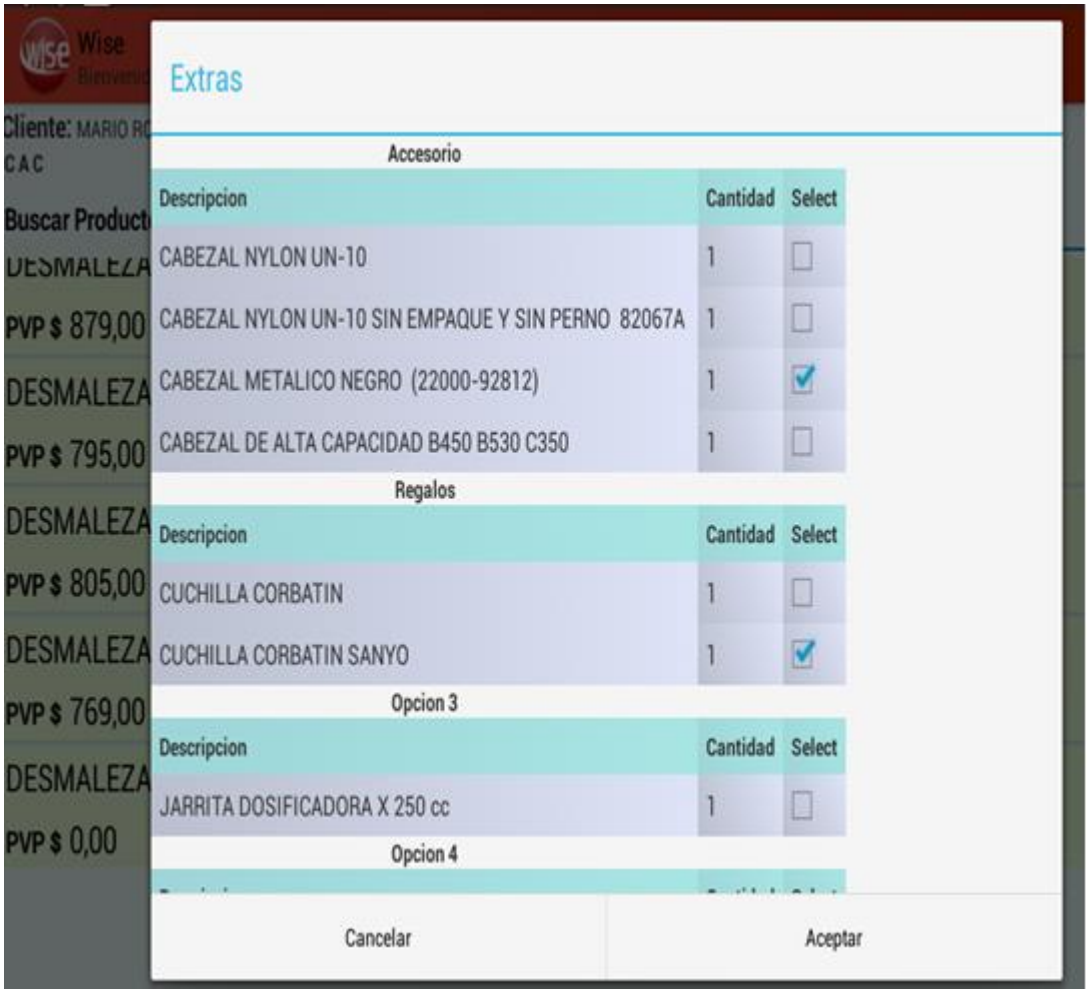


Ilustración 148. Listado de productos agrupados en combo. **Fuente:** El autor.

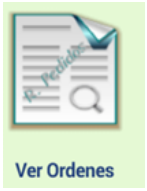
NOTA:

Al intentar seleccionar más de un ítem de cada grupo de extra o combo se emitirá un mensaje de error.

6) Editar el descuento y cantidad del ítem y pulsar el botón *"Grabar"* para hacer efectivo los cambios.

Ilustración 149. Edición del descuento y cantidad**. Fuente:** El autor.

7) Para visualizar las ordenes ingresadas debemos acceder al *"Módulo Pedidos"* y seleccionar el formulario *"Ver Ordenes"*.



8) Seleccionar el rango de fecha para iniciar la búsqueda y pulsar el botón *Generar*. Allí se listarán todas las facturas emitidas en esa fecha detallando el estado del pedido.

Si el estado se encuentra en color verde la orden ha llegado a la central

Ilustración 150. Orden de pedido enviada a la central. **Fuente:** El autor.

Si el estado se encuentra en color rojo la orden todavía está pendiente.

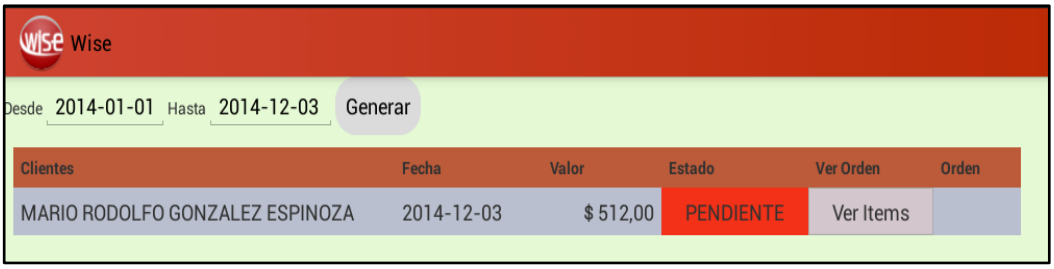


Ilustración 151. Orden de pedido pendiente. **Fuente:** El autor.

9) Si desea ver el detalle de los ítems comprados pulsar el campo *"Ver Items"*

Ilustración 152. Detalle de la orden de pedido. **Fuente:** El autor

- ***Módulo Productos***

Contiene un catálogo categorizado de todos los productos disponibles para la venta.

Al acceder a una opción de categorización se lista los productos agrupados por líneas, categorías, grupos o marca.

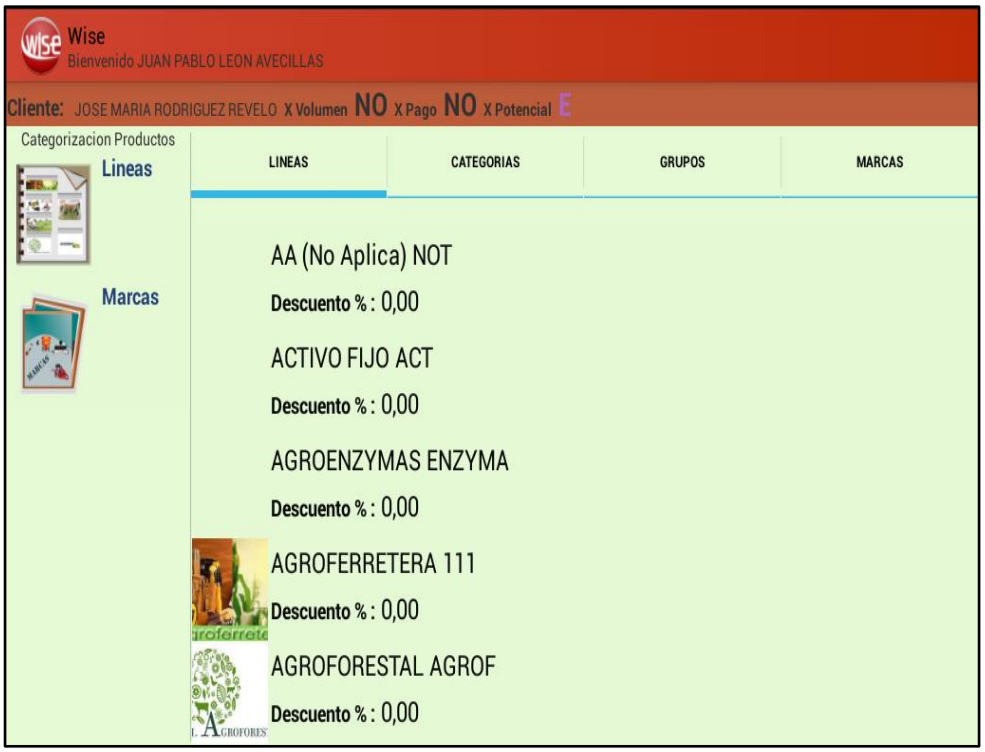


Ilustración 153. Catálogo categorizado de productos. **Fuente:** El autor

Al seleccionar un item se visualizará la siguiente pantalla indicando el detalle del producto, stock, bodega y la imagen. Al momento de pulsar el botón *Agregar Pedido* el item será agregado directamente a la orden.

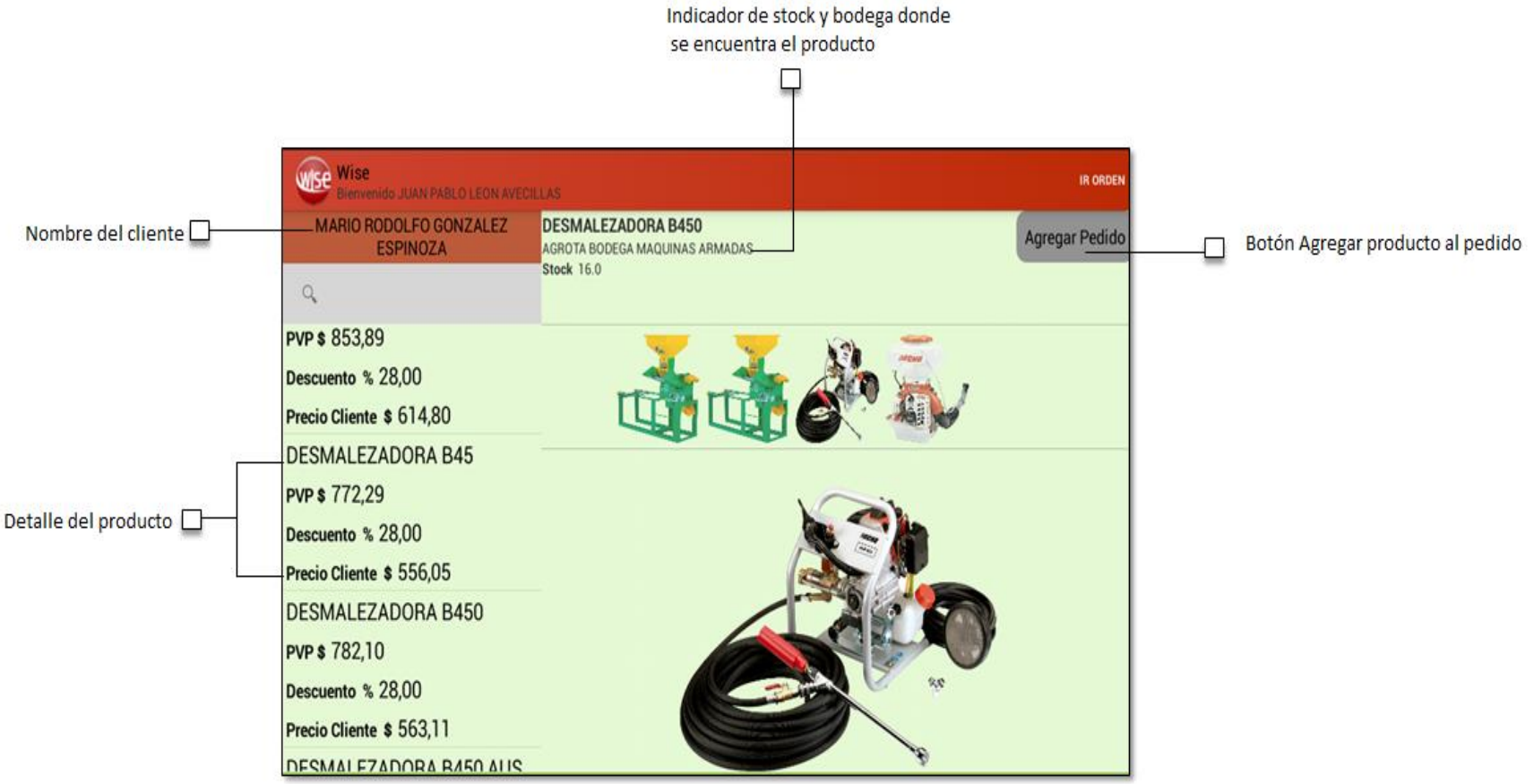
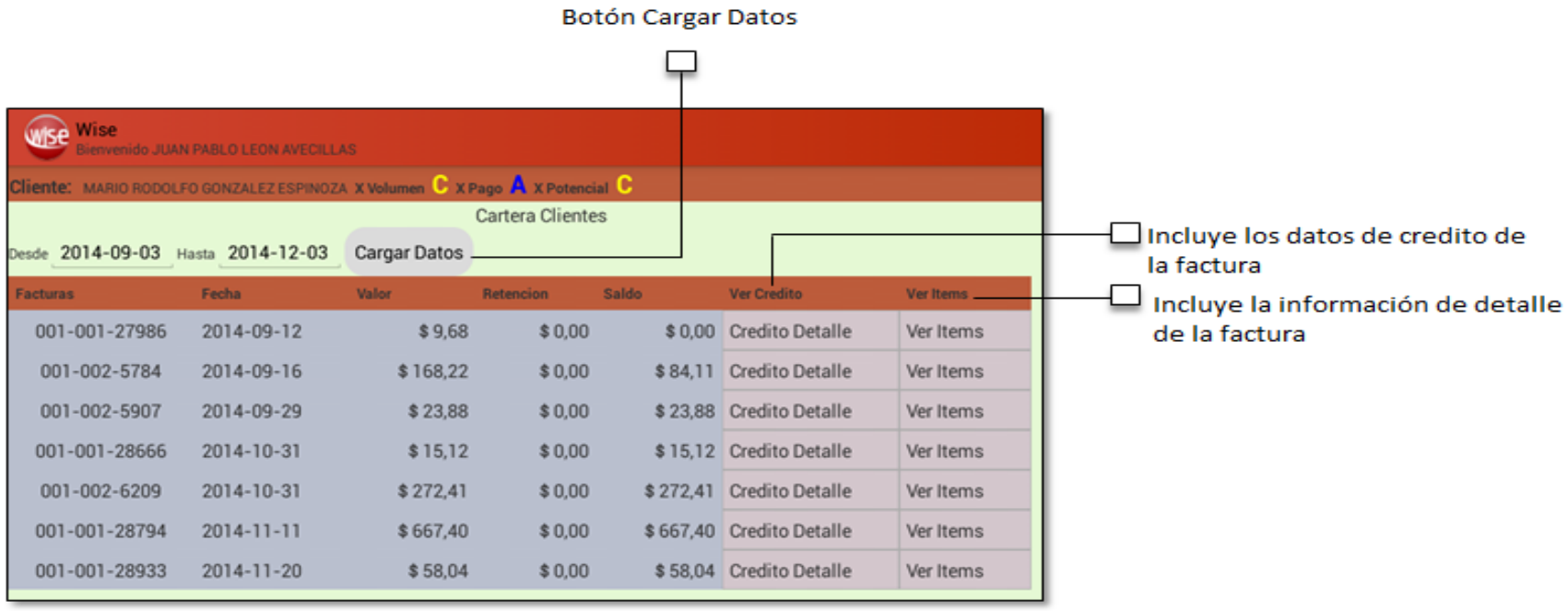


Ilustración 154. Descripción del producto. **Fuente:** El autor

- ***Módulo Cartera***

Permite consultar la cartera del cliente filtrando las facturas emitidas por un rango de fecha.



Facturas	Fecha	Valor	Retencion	Saldo	Ver Credito	Ver Items
001-001-27986	2014-09-12	$ 9,68	$ 0,00	$ 0,00	Credito Detalle	Ver Items
001-002-5784	2014-09-16	$ 168,22	$ 0,00	$ 84,11	Credito Detalle	Ver Items
001-002-5907	2014-09-29	$ 23,88	$ 0,00	$ 23,88	Credito Detalle	Ver Items
001-001-28666	2014-10-31	$ 15,12	$ 0,00	$ 15,12	Credito Detalle	Ver Items
001-002-6209	2014-10-31	$ 272,41	$ 0,00	$ 272,41	Credito Detalle	Ver Items
001-001-28794	2014-11-11	$ 667,40	$ 0,00	$ 667,40	Credito Detalle	Ver Items
001-001-28933	2014-11-20	$ 58,04	$ 0,00	$ 58,04	Credito Detalle	Ver Items

Ilustración 155. Cartera del cliente. **Fuente:** El autor

La opción *Ver Crédito* contiene el detalle de las cuotas, valor, vencimiento, días crédito, días pago, forma de pago, fecha de pago y saldo. Si el cliente ha cancelado su cuota el saldo se presentara en color verde.

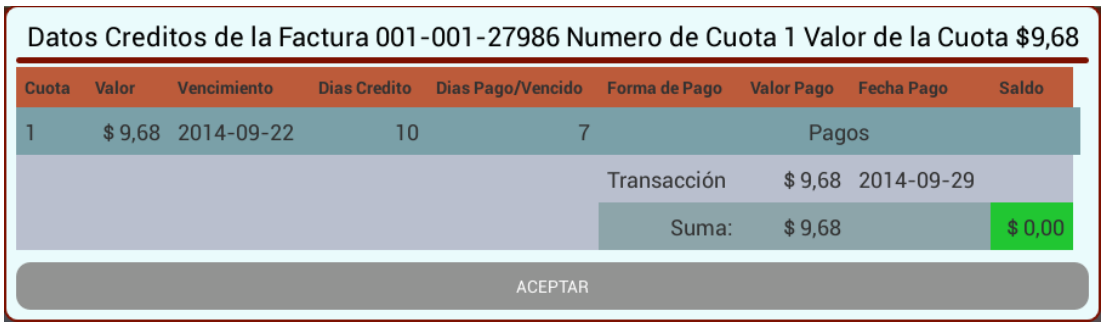


Ilustración 156. Cartera del cliente opción Ver Crédito. **Fuente:** El autor

Caso contario si el cliente tiene todavía cuotas pendientes el saldo se presentará en color rojo.

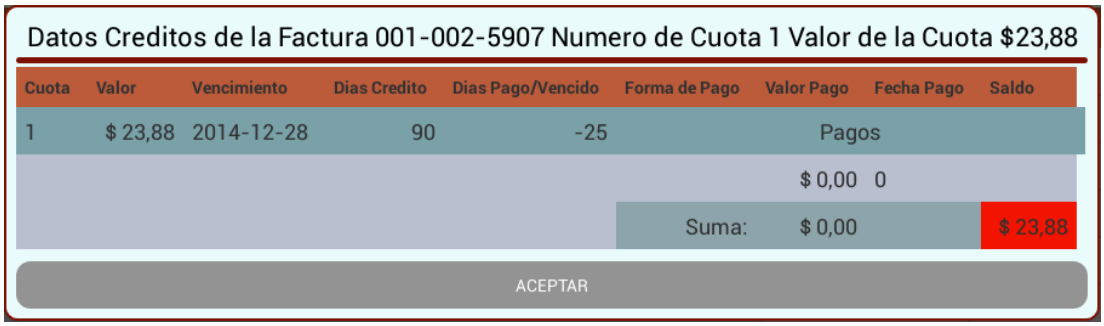


Ilustración 157. Cartera del cliente opción Ver Crédito-Salado pendiente

La opción *Ver Items* nos presenta el detalle de la factura como: items, descripción del item, cantidad, precio y subtotal.

Ilustración 158. Cartera del cliente opción Ver Items. **Fuente:** El autor

- ***Módulo Hoja de Visita***

Permite llevar un registro de la visita realizada al cliente, cobros efectuados, clientes nuevos, ordenes de pedido emitidos además de la información de ayuda sobre la aplicación WISE.

1. ***Registro de visita:*** Sirve como una constancia de la visita realizada al cliente. Aquí debemos elegir el *Estado del cliente*. Por ejemplo: El cliente no está en el local. Luego Ingresar la *Observación* y pulsar el botón *Grabar*.

Ilustración 159. Interfaz Registro de Visita. **Fuente:** El autor

2. ***Cobros:*** Presenta tres formas de cobro ya sea efectivo, cheque o transferencia, al elegir un opción se deberá *Ingresar el Valor de Cobro* y el valor de la *Factura por cobrar* para luego pulsar el botón *Grabar*.

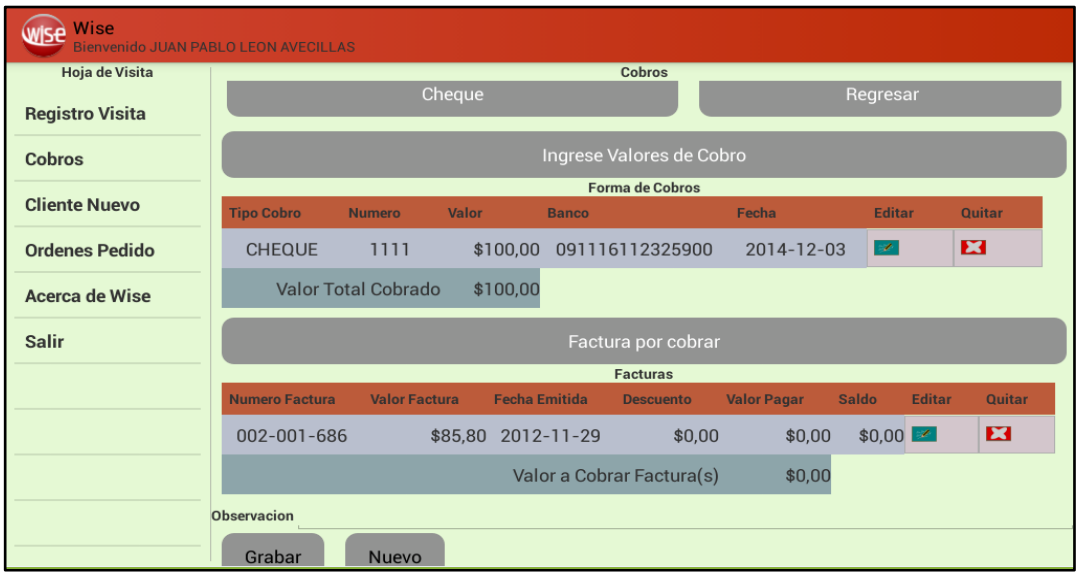


Ilustración 160. Interfaz de Cobros. **Fuente:** El autor

3. ***Cliente Nuevo:*** Se deberá ingresar la información del cliente como: cedula, nombres, apellidos, teléfono, observación y dar click en *Grabar*.

Ilustración 161. Interfaz para el registro de un cliente nuevo. **Fuente:** El autor

4. ***Ordenes Pedido:*** Esta opción redirecciona al módulo pedido permitiendo crear una nueva orden o ver las ordenes emitidas.

Ilustración 162. Interfaz ordenes de pedido. **Fuente:** El autor

- ***Módulo Reportes***

  Genera un reporte sobre los cobros efectuados, clientes nuevos y cartera de los clientes activos.

Ver Cobros: Permite filtrar por fecha las facturas emitidas al cliente indicando el estado, el detalle de la factura y el cobro.

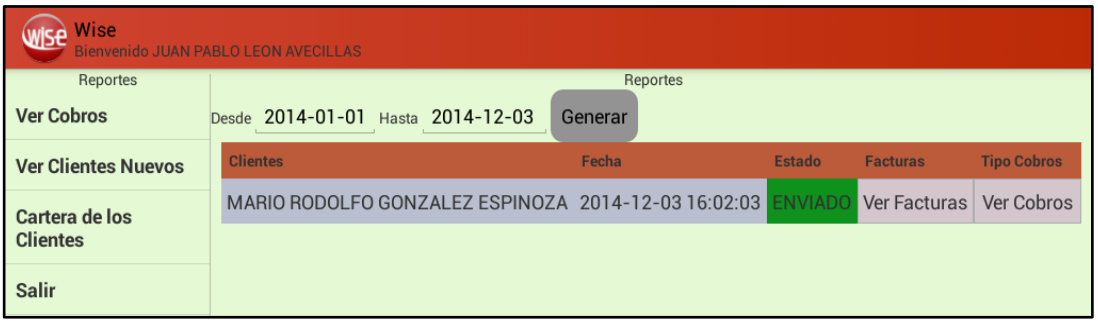


Ilustración 163. Interfaz Ver Cobros. **Fuente:** El autor

Ver Clientes Nuevos: Lista todos los nuevos clientes registrados.
Cartera de los clientes: Lista todos los clientes que disponen de una cartera activa.

Ilustración 164. Interfaz Cartera de los clientes. **Fuente:** El autor

6. CONCLUSIONES

Un modelo de negocio basado en dispositivos móviles implica la ejecución de tareas concretas, cortas y rápidas. Android han demostrado ser lo suficientemente estable para soportar una alta transaccionalidad, disponibilidad y concurrencia de usuarios. A nivel empresarial una aplicación móvil facilita la movilidad del negocio, la comunicación con el cliente además de agregar un plus empresarial sobre la forma en la que se ofertan los productos frente a la competencia.

En una aplicación corporativa donde se maneja una gran cantidad de información, es importante separar las responsabilidades en capas. Considerando tanto la división lógica (asignación de tareas) como la distribución física (asignación de equipos). Al usar un estilo arquitectónico basado en tres capas se optimiza la reutilización de código, el mantenimiento de las reglas de negocio y la detección de fallos.

Al investigar y analizar las diferentes tecnologías que podrían facilitar la integración entre ambos sistemas se optó por la creación de un Web Services, cuya función es compartir la lógica del negocio, datos y los procesos, permitiendo que los datos que residen en el servidor central sean transferidos al dispositivo móvil sin la necesidad de conocer su plataforma o lenguaje de programación. Otro punto a destacar es la escalabilidad ya que al trabajar con un protocolo estandarizado como lo es SOAP se facilita la integración de nuevas tecnologías además de aportar mayor seguridad.

En una aplicación móvil la seguridad es un factor clave dado que implica el intercambio de información a través de un medio inseguro. Las restricciones de seguridad creadas en la aplicación móvil fueron crear un algoritmo personalizado por el desarrollador, el cual permite cifrar y descifrar el mensaje agregando un nivel mayor de seguridad dado que el algoritmo no sigue un patrón conocido como los algoritmos convencionales.

Desde el punto de vista del diseño no es lo mismo crear una aplicación para un teléfono que para una tablet,en realidad son dos medios diferentes tanto en en espacio como en la forma de navegación. La pantalla de un celular limita mucho el diseño de los componentes visuales, en el caso de una tablet los detalles que no eran tan visibles pueden llegar a ser protagonistas con un buen tratamiento de interfaz,es aquí donde los patrones de diseño nos ayudan a organizar y distribuir el área de trabajo creando una interfaz consecuente.Otro aspecto a recalcar es el beneficio de navegación ya que no se requiere programar tantos niveles para llegar a un contenido, interfaces que estaban separadas en un teléfono pueden llegar a tener el mismo nivel en una tablet.

Para terminar podemos decir que para asegurar la calidad del software es aconsejable que el desarrollador se involucre en todas las etapas del proyecto, desde la concepción inicial de la idea (especificación de requerimientos) hasta la aplicación de pruebas.Si bien muchos desarrolladores acostumbramos obviar algunas fases ya sea por tiempo o experiencia, su importancia toma mayor relevancia en proyectos complejos en los cuales la detección de errores,documentación y mantenimiento resulta mucho más sencillo.

7. RECOMENDACIONES

Como recomendación futura para la empresa se plantea que la aplicación sea multiplataforma,es decir utilizar un lenguaje estándar que sirva para cualquier dispositivo móvil independientemente del tamaño, hardware, rendimiento o sistema operativo del dispositivo.Un ejemplo seria usar un híbrido entre la tecnología HTML 5 + Java Script.

También se plantea la incorporación de nuevos servicios que permitan optimizar los componentes del dispositivo, por ejemplo: controles para optimización de bateria, ahorrro de energía, limitar el uso de aplicaciones innecesarias que estuvieran corriendo en segundo plano etc.

Además se propone plantear nuevas funcionalidades como por ejemplo: seguimiento de pedidos que permitan retroalimenar a la empresa sobre la ubicación donde se originó el pedido y la localización del cliente. Asi mismo se recomienda seguir desarrollando aplicaciones en el campo empresarial que permitan tener una comunicación directa con el cliente.

8. BIBLIOGRAFÍA

[1] GIRONES, Jesús Tomás, *El gran libro de Android* , 2da .Edición, Marcombo, Barcelona-España, 2012.

[2] BURNETTE, Ed , *Android:Introducing Googles Mobile Development Platform*, 3ra.Edición, Pragmatic Bookshelf, Barcelona-España, 2008. Disponibleen:http://androidon.ru/books/Hello_Android_Introducing_Google_s_Mobile Development_Platform_December_2008.pdf.

[3] Tipos de sistemas operativos de acuerdo al kernel. Recuperado el 30 de Mayo de 2014. Disponible en:Wordpress.org: http://chsos20121909049.wordpress.com/2012/04/26/tipos-sistemas-operativos-de-acuerdo-al-kernel/.

[4] Universidad Politécnica de Valencia, Diplomado de especialización en desarrollo de aplicaciones Android: Comparativa con otras Plataformas . (Diciembre 16, 2013). Recuperado el 16 de Abril de 2014. Disponible en: http://www.androidcurso.com/index.php/curso-android-basico/tutoriales-android-basico/31-unidad-1-vision-general-y-entorno-de-desarrollo/98-comparativa-con-otras-plataformas.

[5] GARTNER GROUP COMPANY, Mobile Computing. (2013). Recuperado el 17 de Abril de 2014. Disponible en: http://www.gartner.com/technology/mobile/.

[6] ACEVEDO, Jas. "Lo bueno y lo feo del mobile world congress 2014", *PCWORLD Español*,vol 30 , Merliot, 06 de marzo de 2014. Disponible en: http://www.pcworldenespanol.com/2014/03/06/analisis-lo-bueno-lo-malo-y-lo-feo-del-mobile-world-congress-2014/.

[7] FINDLATER,Will. Las grandes conclusiones del Mobile World Congress de Barcelona 2014. [programa televisivo]. CNNMéxico. Disponible en: http://mexico.cnn.com/tecnologia/2014/02/27/las-grandes-conclusiones-del-mobile-world-congress-de-barcelona-mwc2014

[8] STUART,Miles. (Febrero 26, 2014). Los mejores 'smartphones' del Mobile World Congress 2014. [programa televisivo]. CNNMéxico. Disponible en: http://mexico.cnn.com/tecnologia/2014/02/26/los-mejores-gadgets-del-mobile-world-congress-barcelona-2014-mwc2014.

[9] GARCIA,José. (Septiembre 18, 2014). Historia y Versiones Android. Recuperado el 23 de Octubre de 2014.Disponible en:http://javiergarbedo.es/desarrollo-android/80-primeros-pasos/327-historia-y-versiones-de-android.

[10] Condesa. (Julio 04, 2012). Arquitectura de Android. Recuperado el 17 de Abril de 2014. Disponible en: http://androideity.com/2011/07/04/arquitectura-de-android/

[11] TIOBE SOFTWARE QUALITY COMPANY. (Noviembre, 2014). TIOBE Index for November 2014. Recuperado el 30 de Noviembre de 2014. Disponible en: http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html.

[12] NETMARKETSHARE COMPANY. (Octubre, 2014).Market Share Reports. Recuperado el 01 de Diciembre de 2014. Disponible en: http://www.netmarketshare.com/.

[13] MANU, Mateos. Tecnología push así funciona. Recuperado el 01 de Mayo de 2014. Disponible en :http://www.genbeta.com/a-fondo/tecnologia- push-asi-funciona.

[14] MONTERO, Roberto. Mecanismos para el envío de notificaciones a los usuarios de nuestra aplicación. Recuperado el 07 de Mayo de 2014. Disponible en: http://www.javahispano.org/android/2012/6/18/mecanismos-para-el-envio-de-notificaciones-a-los-usuarios-de.html.

[15] SANJIVA ,Weerawarana, y otros, *Web Services Platform Architecture: SOAP*, 1ra .Edición, Prentice Hall, USA, 2005. Disponible en: http://www.bokus.com/bok/9780131488748/web-services-platform-architecture/.

[16] CORTES, José. Simple technologies SOAP. Recuperado el 18 de Mayo de 2014.Disponible en: http://kobjects.org/ksoap2/index.html.

[17] ROBERTO, Carlos. Tres alternativas para crear un catálogo online. Recuperado el 20 de Mayo de 2014. Disponible en: http://www.tecnologiapyme.com/servicios-web/tres-alternativas-para-crear-un-catalogo-online

[18] Catálogos "Page Flip".(2014). Recuperado el 22 de mayo de 2014. Disponible en: ww.kantaronet.com/productos_y_servicios.php?&id=42.&cok=sn&len=1

[19] CATALAN, Adrian. Geolocalización. Recuperado el 23 de Mayo de 2014. Disponible en: http://www.maestrosdelweb.com/editorial/curso-android-geolocalizacion-utilizacion-mapas-google/.

[20] BURCHAT, Michael. Si vas a pasar un montón de horas offline, estas son las 13 aplicaciones para aprovechar el móvil. Recuperado el 05 de Mayo de 2014. Disponible en: http://www.xatakamovil.com/espacio-sony/si-vas-a-pasar-un-monton-de-horas-offline-estas-son-las-13-aplicaciones-para-aprovechar-el-movil

[21] SOLIS, Carlos. Usando aplicaciones offline. Recuperado el 26 de Mayo de 2014. Disponible en: https://developer.mozilla.org/es/Apps/Using_apps_offline.%20Fecha%20de%20consulta:%2005%20de%20Junio%20de%202014.

[22] LUBBERSTEDT, James . DOM Storage – The new super cookie you have dreamed of. Recuperado el 28 de Mayo de 2014. Disponible en:http://webdevwonders.com/dom-storage-super-cookie/.

[23] NAVARRO, Rafael. Modelado, Diseño e Implementación de Servicios Web. Recuperado el 14 de Mayo de 2014. Disponible en: http://users.dsic.upv.es/~rnavarro/NewWeb/docs/RestVsWebServices.pdf

[24] ROBLEDO, Sacristán, y otros, *Programación en Android,* 1ra Edición, Ministerio de Educación de España, España, 2011. Disponible en: http://bibliotecavirtual.ups.edu.ec:2051/lib/bibliotecaupssp/docDetail.action?docID=10779504&p00=aplicaciones+moviles+android.

[25] SANCHEZ, Miguel. (Julio 15, 2013). Gestión de la Seguridad de la Información Coorporativa en dispositivos móviles. Recuperado el 09 de Mayo de 2014. Disponible: http://technologyincontrol2.wordpress.com/2013/07/15/gestion-de-la-seguridad-de-la- informacion-corporativa-en-dispositivos-moviles.

[26] Restricted Profiles. Recuperado el 12 de Mayo de 2014. Disponible en: hpps://support.google.com/nexus/answer/3175031?hl=es-419.

[27] GULLERMO, Julián. Autenticación en dos pasos: qué es, cómo funciona y por qué deberías activarla. Recuperado el 14 de Mayo de 2014. Disponible en: http://www.xataka.com/otros/autenticacion-en-dos-pasos-que-es-como-funciona-y-por-que-deberias-activarla.

[28] La máquina virtual Dalvik. (Julio 07, 2011). Recuperado el 16 de Mayo de 2014. Disponible en: http://www.xataka.com/otros/autenticacion-en-dos-pasos-que-es-como-funciona-y-por-que-deberias-activarla.

[29] DIMAS, José. (Junio 11, 2014). Ciclo de vida de una actividad. Recuperado el 16 de Mayo de 2014.Disponible en: http://www.desarrolloweb.com/articulos/ciclo-vida-actividad- aplicacion.html .

[30] Ciclo de vida de una actividad en Android. (abril 09, 2011). Recuperado el 18 de Mayo de 2014. Disponible en: http://www.terminalesandroid.com/Ciclo-de-vida-de-una-actividad-en-android.

[31] LAGOS, Gonzalo. ¿Por qué es importante tener una tienda virtual?. Recuperado el 21 de Mayo de 2014. Disponible en: http://pymex.pe/emprendedores/comercio-electronico/por-que-es-importante-tener-una-tienda-virtual.

[32] LOPEZ, Miguel. (Agosto 03, 2012). La App Store tiene más de 400.000 aplicaciones que no se han descargado ni una vez, ¿de quién es la culpa?. Recuperado el 22 de Mayo de 2014. Disponible en: http://www.applesfera.com/aplicaciones-ios-1/la-app-store-tiene-mas-de-400-000-aplicaciones-que-no-se-han-descargado-ni-una-vez-de-quien-es-la-culpa.

[33] SOTO, Beatriz. 20 aplicaciones Android para autónomos o empresas. Recuperado el 25 de Mayo de 2014. Disponible en: http://www.gestion.org/gestion-tecnologica/nuevas-tecnologias/38750/20-aplicaciones-android-para-autonomos-o-empresas.

[34] *Hid Activo Soft Token*. Recuperado el 28 de Mayo de 2014. Disponible en: www.pandaid.com/hid-activid-soft-token/.

[35] KENDALL, Kennethe y KENDALL , Julie E , *Análisis y diseño de sistemas*, 8va .Edición, Pearson Educacación , México, 2011. Disponible en: http://www.academia.edu/7102592/Analisis_y.Diseno_de_Sistemas_8ed_Kendall_PDF

[36] SANCHEZ, María. Herramientas de geolocalización: qué son y para qué sirven. Recuperado el 05 de Junio de 2014. Disponible en: http://creatic.innova.unia.es/otros/geolocalizacion

[37] VITTONNE, José y COELLO, Javier, *Diseñando apps para móviles*, 1ra .Edición, José Vittone, Barcelona -España, 2013. Disponible en : http://www.appdesignbook.com/es/contenidos/presentacion/

[38] Bases de datos portables con SQLite. Recuperado el 18 de Junio de 2014.Disponible en: http://3palmeras.wordpress.com/2012/10/07/bases-de-datos-portables-con-sqlite/

[39] ALAIN , Abran , y otros , *SWEBOK: Guide to the software engineering Body of Knowledge* , IEEE, 2004, p.202.

[40] ¿Qué es la Usabilidad? . Recuperado el 20 de Junio de 2013. Disponible en: http://www.guiadigital.gob.cl/articulo/que-es-la-usabilidad

[41] SOMERVILLE, Ian, *Ingenieria del Software* , 7ma .Edición, PEARSON EDUCATION, Madrid- España, 2005. Disponible en: http://es.slideshare.net/jasc_584/ingenieriadesoftware-iansommerville7maedicion-9417118

[42] IEEE, Std 29148 *Requirements Engineering* .The Institute Of Electrical and Electronics Engineers, 2011, p.95.

[43] PRESSMAN, Roger, *Ingenieria del Software un Enfoque Práctico*, 5ta Edición, McGRAW- HILL INTERAMERICANA DE ESPAÑA S.A , Madrid-España, 2002.

[44] MITTAL, Sumit. Checking Network Connection Android. Recuperado el 18 de Noviembre de 2014. Disponible en: http://www.tutorialspoint.com/android/android_network_connection.htm

[45] Geolocalización. Recuperado el 18 de Noviembre de 2014. Disponible en: ULTRA-LAB // CODE: http://ultra-lab.net/code/wiki:gps

[46] Paper: KRUCHTEN, Philippe, Architectural Blueprints—The "4+1" View , *IEEE Software, 12*(6), p.42-50.

[47] Arquitectura tres niveles. Recuperado el 12 de junio de 2014. Disponible en: http://www.soluciones-del-valle.net/archivos/Tecnologico/antologias/ene-jun-13/li8/Arquitectura_de_3niveles.pdf

[48] VILLALOBOS, Jesús , Arquitectura de Aplicaciones Distribuidas. Recuperado el 17 de Septiembre de 2014. Disponible en:http://knol.google.com/k/aplicaciones-distribuidas#

[49] Principios de Diseño Orientado a Objetos. Recuperado el 01 de Agosto de 2014. Disponible en:http://www.escet.urjc.es/~emartin/docencia0809/poo/2.-DisenoOO-SinSols-4p.pdf

[50] GONZALEZ, José , *¿Qué es UML?*. Recuperado el 03 de Agosto de 2013.Disponible en: http://www.docirs.com/uml.htm

[51] Diseñar Apps Android, buenas prácticas y patrones. Recuperado el 05 de Agosto de 2014. Disponible en: http://www.muchoandroid.com.ar/2012/04/disenar-apps-android-buenas-practicas-y.html

[52] Action Bar. (2014) . Recuperado el 08 de Agosto de 2014. Disponible en: http://developer.android.com/design/patterns/actionbar.html

[53] ARANA, Rayco (Abril 29, 2012). Diseñar Apps Android, buenas practicas y patrones. Recuperado el 13 de Agosto de 2014. Disponible en: http://www.muchoandroid.com.ar/2012/04/disenar-apps-android-buenas-practicas-y.html

[54] TRISTAN, José , Manejo de datos en Android: SQLITE. Recuperado el 05 de Noviembre de 2014. Disponible en : en:http://www.javahispano.org/android/2011/12/27/manejo-de-datos-en-android-sqlite.html

[55] TAPIA, José . (Noviembre,2012), Tutorial SQLite en español. Recuperado el 05 de Octubre de 2014. Disponible en: http://usemossoftwarelibre.wordpress.com/cc/tutorial-sqlite-en-espanol/

[56] ROSALES, Marianela , Arquitectura Orientada a Servicios. Recuperado el 08 de Noviembre de 2014. Disponible en: http://ccia.cujae.edu.cu/index.php/siia/siia2008/paper/viewFile/1186/255

[57] BIADA, Toni , Creación de aplicaciones mobile offline: Estrategia de desarrollo apps offline first. Recuperado el 17 de Noviembre de 2014. Disponible en: http://www.einnova.com/notas/servicio/creacion-de-aplicaciones-offline

[58] Página oficial Oracle. Disponible en: http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html

[59] Página oficial Android SDK . Disponible en: http://developer.android.com/sdk/index.html

[60] Complejidad ciclomática Descripción y Aplicaciones. Recuperado el 22 de Noviembre de 2014. Disponible en: http://centrodeartigo.com/revista-digital-universitaria/contenido-40474.html

[61] TUYA, Javier, y otros, *Técnicas Cuantitativas para la Gestión en la Ingeniería del Software*, 1ra .Edición, Gesbiblo S.L , La Coruña - España , 2007.

[62] FERNANDEZ, Juan Manuel, Caminos Básicos. Recuperado el 01 de Diciembre de 2014. Disponible en: http://www.uv.mx/personal/jfernandez/files/2010/07/Cap3-Caminos.pdf

[63] IEEE, Stda 10160 Standard for Information Technology System Desing. The Institute Of Electrical and Electronics Engineers*, 2009, p.40.*

9. ANEXOS

Sección ISO/IEC 29148-2011	Sección ISO/IEC 29148	ID	Requerimiento	ID Caso de Uso
2.1 Identificación de los Requerimientos: Comprende todas las tareas necesarias para satisfacer las necesidades de los involucrados.	**2.1.2.1** Requerimientos Funcionales	**RF-001**	Ingresar al sistema	**CU-001**
2.1 Identificación de los Requerimientos: Comprende todas las tareas necesarias para satisfacer las necesidades de los involucrados.	**2.1.2.1** Requerimientos Funcionales	**RF-002**	Actualizar datos servidor central.	**CU-002**
2.1 Identificación de los Requerimientos: Comprende todas las tareas necesarias para satisfacer las necesidades de los involucrados.	**2.1.2.1** Requerimientos Funcionales	**RF-003**	Listar Cliente	**CU-003**
2.1 Identificación de los Requerimientos: Comprende todas las tareas necesarias para satisfacer las necesidades de los involucrados.	**2.1.2.1** Requerimientos Funcionales	**RF-004**	Registrar Orden de Pedido.	**CU-004**
2.1 Identificación de los Requerimientos: Comprende todas las tareas necesarias para satisfacer las necesidades de los involucrados.	**2.1.2.1 Requerimientos Funcionales**	**RF-005**	Listar Productos	**CU-005**
2.1 Identificación de los Requerimientos: Comprende todas las tareas necesarias para satisfacer las necesidades de los involucrados.	**2.1.2.1** Requerimientos Funcionales	**RF-006**	Agregar productos por catálogo a la Orden de Pedido.	**CU-006**
2.1 Identificación de los Requerimientos: Comprende todas las tareas necesarias para satisfacer las necesidades de los involucrados**.**	**2.1.2.1** Requerimientos Funcionales	**RF-007**	Listar Orden de Pedido.	**CU-007**
2.1 Identificación de los Requerimientos: Comprende todas las tareas necesarias para satisfacer las necesidades de los involucrados.	**2.1.2.1** Requerimientos Funcionales	**RF-008**	Listar Cartera	**CU-008**
2.1 Identificación de los Requerimientos: Comprende todas las tareas necesarias para satisfacer las necesidades de los involucrados.	**2.1.2.1** Requerimientos Funcionales	**RF-009**	Actualizar Cartera	**CU-009**
2.1 Identificación de los Requerimientos: Comprende todas las tareas necesarias para satisfacer las necesidades de los involucrados.	**2.1.2.1** Requerimientos Funcionales	**RF-010**	Registrar Hoja de Visita	**CU-010**

Tabla 57.ANEXO A- Especificación de Diseño de Casos de Uso [63].