Universidad Internacional de La Rioja (UNIR)

Escuela de Ingeniería

Grado en Ingeniería Informática

Módulo "Gestión de Almacén" en Android para Xgestevo

Ubicación del código fuente:
https://sourceforge.net/projects/xalmacen/

Trabajo Fin de Grado

Presentado por: Aguado Rodrigo, Manuel David

Director/a: Cobos Guzmán, Salvador

Ciudad: Cartagena
Fecha: 7 de Julio de 2017

Resumen

Este Trabajo Fin de Grado (TFG) presenta la creación de un módulo de control de almacén para el software de Planificación de Recursos Empresariales (Enterprise Resource Planning o ERP) Xgestevo.

Dicho módulo de almacén se hará desde cero, sin contar con nada de código original del propio software de gestión, accediendo y manipulando directamente la base de datos MySQL que almacena la información con la que trabaja el software.

La plataforma elegida para dicho módulo será Android. La gran adaptabilidad de esta plataforma a los distintos formatos y dispositivos del mercado, su integración total con los terminales que la ejecutan y la rápida sincronización con diversos periféricos, la hacen la mejor elección.

Este software constará de tres módulos principales: Control de stock, entrada de material y preparación de pedidos. Además, constará de sistemas de identificación y de configuración de conexiones.

Palabras Clave: Android, Xgestevo, movilidad, gestión, almacén

Abstract

This End of Grade Work presents the creation of a warehouse control module for Xgestevo Enterprise Resource Planning (ERP) software.

This module will be programmed from nothing, without taking notice of the own software original code, accessing and directly manipulating the MySQL database which stores the information needed for the software.

The chosen platform for this module is Android. The great adaptability of this platform to the different formats and devices currently in the market, it totally inclusion in the devices that execute it and it rapid synchronization with different peripherals, make it the best choice.

This software will be formed by three main modules: stock control, goods receipts and orders preparation. In addition, it will have a login and a connection configuration system.

Keywords: Android, Xgestevo, mobility, management, warehouse

Índice de contenido

Índice de ilustraciones

Índice de tablas

1. Introducción

Xgestevo (Saura, D. ,2015) es un sistema ERP que, desde su nacimiento en 1998, ha ido creciendo y evolucionando con las empresas que lo usan. Son muchos los módulos que componen este software: gestión de clientes y proveedores, logística, inteligencia de negocio, gestión documental, Servicio de Asistencia Técnica (SAT a partir de ahora) a clientes y Autorizaciones de Devolución de Mercancía a proveedores (Return Material Authorization o RMA a partir de ahora) entre otros. Pero, el que interesa para el desarrollo de este proyecto, apareció a principios de 2009 y se llama Regumobile.

Regumobile es un módulo de control de almacenes que está programado para terminales con Windows CE 5. El uso de esos terminales fue un gran avance para el trabajo del personal de los almacenes que usaban Xgestevo, pues les permitía ahorrar muchos viajes al terminal fijo, para consultar datos de stock y ubicaciones. También aceleró el proceso de preparación de pedidos, pues el personal ya no necesitaba ir a buscar uno a uno los productos y llevarlos a donde estuviera el terminal que usaba Xgestevo para hacer el albarán y mandar el pedido, sino que ahora podían llevar cargado en el terminal móvil todos los datos del pedido e ir recogiendo todos los productos sin necesidad de pasar por un único terminal fijo. Poco después también aparecería una nueva funcionalidad para poder también hacer la entrada de material de manera que, no se tuvieran que cargar grandes cajas y palés de un sitio a otro del almacén para acercarlo a los terminales fijos, sino que ahora se desplazaba el personal del almacén a la ubicación del material y directamente lo hacía aparecer en el sistema sin grandes esfuerzos.

Regumobile junto con el servidor de movilidad que lo gestiona, proporcionaron un gran ahorro de espacio, pues ya no eran necesarios terminales fijos con sus periféricos y mesas donde colocarlos. Ahorró mucho tiempo, pues los pedidos, entradas y controles de inventario se realizaban más rápido, no teniendo que consultar continuamente en puestos fijos. Duplicó la producción pues ya no hacía falta gente preparando material y gente haciendo albaranes, o gente acercando grandes bultos para que el personal del puesto fijo hiciera la entrada, sino que todo el esfuerzo se podía dedicar a la preparación en un único paso.

En general optimizó todos los procesos del almacén en una época, 2009, en la que todo esfuerzo por ahorrar costes era poco. Y aunque la inversión en terminales era alta, se asumía como un gasto único que se recuperaba ampliamente a los pocos meses.

Como ya se ha dicho, poco a poco Xgestevo ha ido evolucionando, pero, aunque Regumobile también ha ido añadiendo mejoras, su evolución ha sido menor. Parte de esta

poca evolución viene del entorno elegido en su momento para el desarrollo, Windows Mobile. El propio Microsoft deja obsoleto al poco tiempo la versión usada para el desarrollo, siendo incompatible con las nuevas versiones. Actualmente Windows Mobile es un sistema en desuso y cada vez es más difícil la obtención de terminales para la ampliación o incluso reposición de roturas.

Surge entonces la necesidad de un nuevo entorno, y Android es el gran candidato, pues es un sistema que actualmente está muy asentado, tiene prácticamente dominado el sector de los Smartphones, todos los días salen nuevas opciones, apps, recursos y ayudas para el sistema. Y, sobre todo, los terminales son muy asequibles para cualquier usuario, además de permitir pantallas mucho más grandes, en distintas formatos y orientaciones y con terminales mucho más potentes.

Una vez confirmada la necesidad de este nuevo software y establecido el entorno sobre el que debe funcionar, se plantea este proyecto.

A lo largo de las páginas de esta memoria, se hará un breve resumen tanto de Android, como de Xgestevo y de las distintas alternativas que hay para el control de un almacén que usa este software ERP.

Se plantearán todos los requisitos que harán de este software un buen sustituto y las mejoras en las funcionalidades que en la actualidad son necesarias para un almacén moderno. Además, se establecerá el modelo de desarrollo incremental como metodología de desarrollo, ya que se adapta a la perfección a la creación de un proyecto de una sola persona y cuyos módulos están claramente definidos junto con la dependencia incremental que existe entre ellos.

Se hará una descripción completa de las funcionalidades que se han establecido en los requisitos y posteriormente se dará la explicación técnica de como se ha implementado cada uno de los módulos y sus clases.

Se finalizará con la explicación de las pruebas realizada, que darán lugar a unas conclusiones finales y al punto de partida para nuevas líneas de trabajo o mejoras.

2. Estado del Arte

En este apartado se describirán las dos plataformas principales sobre las que se sustenta este proyecto fin de grado, Android y Xgestevo.

Android será la plataforma de desarrollo y los dispositivos finales a los que está destinado este módulo usarán este sistema. Se dará un breve repaso por su historia, hablando sobre su arquitectura y en general, se verá cómo poco a poco y versión a versión ha ido creciendo hasta dominar prácticamente por completo el mercado de los terminales móviles, tanto Smartphone como Tablet.

Xgestevo es el software principal para el que este módulo hará de apoyo y mejora de uno de sus componentes opcionales. A lo largo de este apartado se podrá ver una descripción del software Xgestevo, también su historia muy brevemente, junto con la evolución a lo largo de algunas de sus versiones. Por último, su estructura, sustentada en una base de datos MySQL o MariaDB y algunos de sus módulos opcionales principales.

Con estos dos repasos generales a las arquitecturas con las que se debe trabajar, se estará en disposición para abordar los demás apartados del proyecto, ya que se tendrá un poco más de conocimiento de la base sobre la que debe construirse el módulo de control de almacén.

2.1 Android

2.1.1 ¿Qué es Android?

Como puede extraerse de Tomás, J. (2017). Android es un Sistema Operativo basado en Linux (Linux, 2016) y de código abierto creado en 2005 por Android Inc. Fue diseñado principalmente para su uso en terminales móviles de pantalla táctil, aunque actualmente se encuentra integrado en multitud de hardware además de tabletas y teléfonos, como pueden ser: relojes, televisores, electrodomésticos y gran cantidad de sistemas embebidos.

Sus aplicaciones son desarrolladas en Java (Java, 2017) y con la interfaz de usuario diseñada en XML, por lo que gracias a la máquina virtual que ejecutan todos los sistemas Android, permiten que sea ejecutable en cualquier CPU presente y futura.

2.1.2 Breve historia

En octubre de 2003 nace Android Inc. en Palo Alto, California, una startup que pretendía crear un sistema operativo para móviles basado en Linux. 22 meses después de su creación, en agosto de 2005, Android Inc. es comprada por Google (Elgin, B. ,2005).

En 2007 Google, junto con algunas de los mayores fabricantes de teléfonos móviles (Motorola, T- Mobile, Samsung, Ericsson, Toshiba, Vodafone) y de componentes (Intel, Texas Instruments), crean la **Open Handset Alliance** (Open Handset Alliance, 2007) cuyo principal objetivo será la de definir los estándares de Android**.** Y el 12 de noviembre de ese mismo año anuncian el lanzamiento del primer SDK (Software Development Kit) para que los desarrolladores comiencen a crear aplicaciones para el nuevo sistema Android.

En febrero de 2008, Reardon, M. (2008) informa de la aparición en Barcelona, en el MWC (Mobile World Congress), de los primeros prototipos con Android. Pero no será hasta octubre de ese año cuando salga a la venta el primer terminal con este sistema, el HTC Dream, también conocido como T-mobile G1 o Google Phone, cuya presentación se podía ver en HTC (2008). Con el lanzamiento del primer terminal con Android, Open Handset Alliance (2008) anuncia que Google hace público el código fuente bajo licencia de código abierto Apache, momento a partir del cual, todos los fabricantes tienen permiso para usar y modificar el sistema sin necesidad de pagar royalties. También aparece este año Android Market.

En el último trimestre de 2010 Android es noticia otra vez, esta vez en EuropaPress de la mano de Portaltic/EP. (2011). Android ha logrado, en esos últimos tres meses, superar a todos sus competidores y se sitúa como el sistema operativo para móviles más usado, rebasando a Symbian de Nokia.

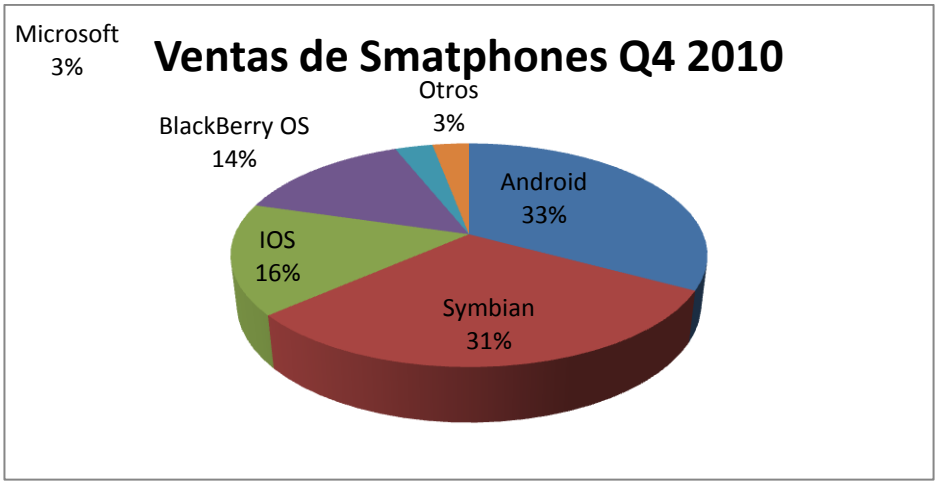

Ilustración 1. Ventas de Smarphones por Sistema Operativo en 4º trimestre de 2010

En 2011 sale la versión 3.0 de Android, es una versión solo para tabletas, con lo que la compañía da otro salto tecnológico y amplía su mercado. Poco después sacará ese mismo año la versión 4.0 que ya está preparada para las dos plataformas, tabletas y smartphones.

2012 es el año en el que aparece Google Play Store, la tienda de aplicaciones que actualmente se conoce y que surge de la fusión de Android Market con Google Music.

En el siguiente grafico se representa la evolución de ventas a las que se ha visto sometido el mercado desde que salió a la venta el primer teléfono con sistema Android.

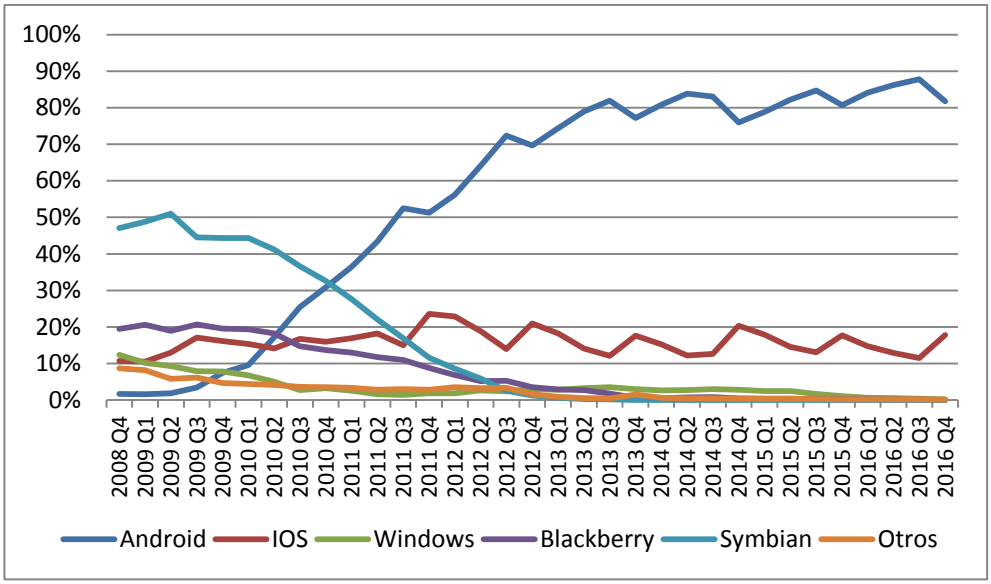

Ilustración 2. Porcentaje de ventas mundiales de Smartphone por SO

En 2008 Q4, trimestre en que sale a la venta el T-Mobile G1, Gartner, (2009) aún no tiene datos reales del nuevo sistema operativo de Google, pero estima que el 20% de las ventas que publica para ese trimestre de teléfonos con Linux, llevaban el sistema Android. Aun no se preveía al enorme impacto que tendría este SO y lo englobaron en las estadísticas con los demás derivados de Linux.

Se aprecia gracias a los informes cuatrimestrales que se van publicando en Gartner, como Android desde su nacimiento ha ido incrementando su presencia continuamente, hasta dominar el mercado de la telefonía móvil: Gartner. Mayo (2010), Gartner. Agosto (2010), Gartner. Noviembre (2010), Gartner. (2011), Gartner. Mayo (2012), Gartner. Agosto (2012), Gartner. Noviembre (2012), Gartner. Febrero (2013), Gartner. Mayo (2013), Gartner. Agosto (2013), Gartner. Diciembre (2014), Gartner. Marzo (2015), Gartner. Mayo (2015), Gartner. Agosto (2015), Gartner. Mayo (2016), Gartner. Agosto (2016), Gartner. Noviembre (2016).

Con los datos oficiales más actualizados de los que se dispone públicamente en Gartner (2017), tras el cierre del año 2016, Android se sitúa con una cuota de mercado del 82% de las ventas mundiales, seguido de IOS con un casi 18% y dejando con menos de un 0,5% del mercado al resto de sistemas.

2.1.3 Versiones de Android

Desde su lanzamiento, Google ha ido presentando un gran número de versiones de Android. Hasta el momento, se dispone ya de la versión 7.0 del sistema.

Una de las características más importantes que se encuentra en las versiones de Android es que tienen una retro compatibilidad total con sus versiones anteriores. Todas las versiones de Android pueden ser consideradas como actualizaciones de las anteriores, solo añaden nuevas funcionalidades y mantienen todas las anteriores, aun cuando se modifican o se consideran obsoletas, simplemente quedan marcadas como *deprecated*, pero jamás son eliminadas, por lo que los desarrolladores pueden seguir usándolas.

El sistema de versiones de Android se ha identificado de tres formas alternativas:

1. **Número de versión**
2. **Nivel de API**, muy útil para los programadores. Identifica de manera incremental con un número entero, el nivel de desarrollo del repertorio de instrucciones. Las aplicaciones se crean para un nivel de API mínimo, a partir del cual se tienen todas las funcionalidades necesarias para que la aplicación funcione correctamente.
3. **Nombre comercial.** Cada versión de Android recibe una inicial desde A y a partir de esa inicial se le da un nombre comercial que empieza con esa letra y pertenece a un nombre de un dulce.

A continuación, se presenta una tabla con las distintas versiones y sus principales características extraídas tanto de Tomás, J. (2017) como de Google. Android (2017):

Tabla 1. Versiones de Android

API	Nombre	Ver.	Fecha	Características
1	Sin nombre	1.0	Sep. 2008	- Versión inicial de Android - Incluido en HTC DREAM
2	Sin nombre	1.1	Feb. 2009	- Corrección de errores
3	Cupcake 	1.5	Abr. 2009	- Teclado en pantalla y widgets - Capacidad de grabar audio y video - Soporte para Bluetooth
4	Donut 	1.6	Sep. 2009	- Búsqueda Avanzada en dispositivo - Gestures y síntesis de texto a voz - Apps soportan distintas resoluciones - Resolución de pantalla WVGA - Soporte CDMA/EVDO, VPN, 802.1x

5	Éclair		2.0	Oct. 2009	- Soporte Bluetooth 2.1 - Manejo centralizado de cuentas - Aumento de pantalla y resolución - Soporte HTML5 - MotionEvent acepta multi-touch
7	Éclair		2.1	Ene. 2010	- Reconocimiento de voz - Fondos de pantalla animados - WebKit maneja BBDD en internet
8	Froyo		2.2	May. 2010	- Gran optimización de velocidad - Nuevo compilador JIT de Dalvik - Adobe Flas 10.1 y Javascript V8 - Instalar Apps en medio externo - Auto Update y Backups en Apps - Compartir Internet (tethering) - Soporte Wifi IEEE 802.11n - OpenGL 2.0
9	Gingerbread		2.3	Dic. 2010	- Resolución WXGA y superior - Nueva interfaz de usuario - Soporte para segunda cámara - Recolector de basura de Dalvik - Sistema de ficheros ext4 - Soporte VOIP/SIP - Tecnología NFC y sensores nativos - Librerías de compatibilidad (Google. Android developer, compatibilidad,2017)
11	Honeycomb		3.0	Feb. 2011	- Versión optimizada para tabletas - Se introducen los Fragments - Soporte procesadores multinucleo - Aparición de la barra de sistema - Motor gráfico Renderscript
12	Honeycomb		3.1	May. 2011	-Protocolos de conexión USB: Transferencia fotos/videos (PTP/MTP) Transferencia en tiempo real (RTP)
13	Honeycomb		3.2	Jul. 2011	- Optimización tamaño de pantallas - Compatibilidad Zoom en Apps - Sincronización multimedia desde SD

14	Ice Cream Sandwich 	4.0	Oct. 2011	- 1ª versión para tablet y smartphone - Gestor de uso de datos - Interfaz de inicio personalizable - Compartir contenido en Android Beam
15	Ice Cream Sandwich 	4.0.3	Dic. 2011	- Inclusión de gestión de redes sociales - Mejoras en calendario, texto a voz y Bases de datos
16	Jelly Bean 	4.1	Jul. 2012	- Mejora rendimiento interfaz de usuario - Nuevas notificaciones accionables - Aparece Google Now
17	Jelly Bean 	4.2	Nov. 2012	- Sistemas multiusuario (solo tabletas) - Photo Sphere (fotos en 360º) - Conexión con TVHD (miracast)
18	Jelly Bean 	4.3	Jul. 2013	- Perfiles restringidos (solo tabletas) - Soporte Bluetooth Low Energy - OpenGL ES 3.0
19	KitKat 	4.4	Oct. 2013	- Optimización para uso con poca RAM - Sistema de pantalla completa - Aparecen los WebView - Sistema de control por voz: OK Google - Nueva máquina virtual ART
21	Lollipop 	5.0	Nov. 2014	- Compatible Android Wear, Tv y Auto - Nuevo enfoque visual, Material Design - Sensor cardiaco, inclinación y gestos - Nuevo modo bloqueo anti-interrupción - OpenGL ES 3.1
22	Lollipop 	5.1	Mar. 2015	- Soporte para varias tarjetas SIM
23	Marshmallow 	6.0	Oct. 2015	- Nuevo administrador de permisos - Copias de seguridad automáticas - Google Now evoluciona a Now on Tap - Autentificación por Huella Digital - Google Pay - Nuevo gestor de batería Doze - Uso de memoria Externa como Interna

24	Nougat 	7.0	Jul. 2016	- Varias aplicaciones en pantalla - Compilación mixta: AOT y JIT - Nueva API de gráficos 3D Vulkan - Plataforma Realidad Virtual Daydream - Mejoras en el ahorro de batería
25	Nougat 	7.1	Dic. 2016	- Acceso directo a Apps personalizable - Inserción de imágenes desde teclado

2.1.4 Estructura de Android

A continuación, se describe la estructura sobre la que se asienta todo sistema Android y para ello lo primero será resumirla en el siguiente grafico que aparece en Tomás, J. (2017), para seguidamente ir describiendo uno a uno los principales módulos y sus características descritos también en Google. Android developer, plataforma (2017)

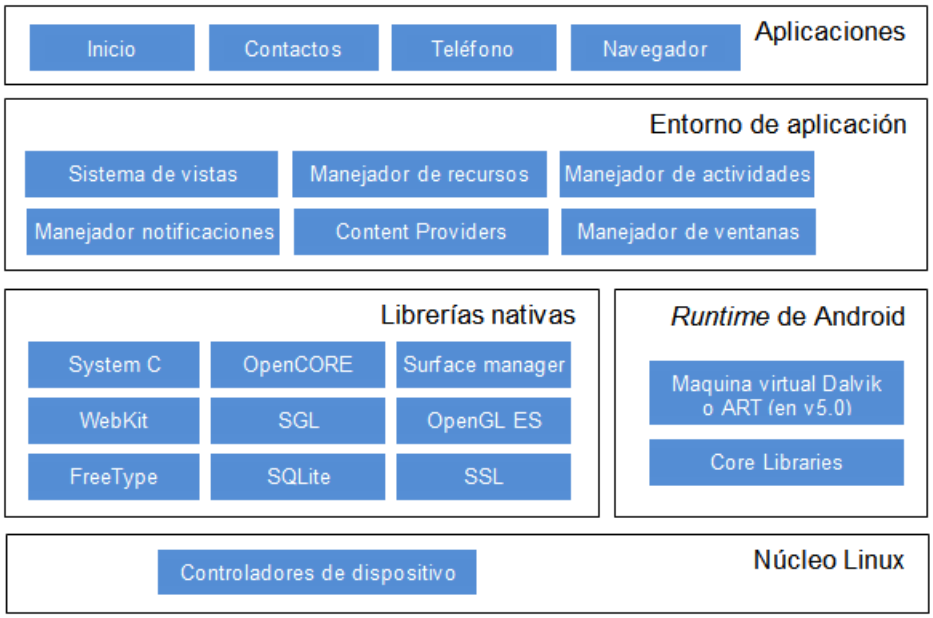

Ilustración 3. Arquitectura de Android

Como puede verse, Android se basa en una pila de cuatro capas de software libre. Estos son los niveles que se describirán a continuación.

El núcleo

El núcleo de Android está basado en kernel de Linux 2.6, gracias a lo cual, Android se puede aprovechar de los sistemas de seguridad, gestión de procesos múltiples, la gestión de memoria y gestión de drivers. Además, al estar basado en un sistema abierto, permite a los desarrolladores trabajar con un sistema lo suficientemente conocido.

Por encima del núcleo existe la capa de abstracción del Hardware o HAL, que aísla el hardware de la API de JAVA. Son un conjunto de módulos que brindan una interface de comunicación con un componente específico, sin necesidad de tener que conocer exactamente el funcionamiento de este.

Runtime

Android basa la ejecución de sus aplicaciones en el concepto de máquina virtual que instauró Java. Pero, debido a los dispositivos a los que el sistema va orientado, se ve limitado por unas prestaciones de memoria y CPU que un ordenador normal no sufre.

Para solventar el problema de la limitación del hardware, Google inventa una máquina virtual propia, Dalvik, que está optimizada para el uso de pocos recursos y ejecuta cada aplicación como un proceso independiente, con su propio hilo e incluso con su propia instancia de Dalvik. Unas de las características de funcionamiento de Dalvik fue que toda aplicación se compilaba en tiempo de ejecución, por lo que generaba un gran gasto de recursos.

A partir de Lollipop (Android 5.0) aparece de manera estándar una nueva máquina virtual más eficiente, ART. Trae sobre todo una mejora sustancial en los tiempos de ejecución y en el ahorro de recursos y energía, pero ¿cómo lo logra? Introduce el concepto de ahead-of-time (AOT) que, en resumen, no es más que la compilación de la aplicación en el momento de su instalación. ART puede usar los dos tipos de compilación y adecuar cada aplicación al uso de una u otra, o incluso compilar ciertos componentes en la instalación y otros en ejecución. También introdujo un sistema de colector de basura mejorado y varios cambios en el sistema de depuración de aplicaciones.

Además de las máquinas virtuales, se incluyen también la gran mayoría de las librerías de JAVA en el Core Libreries.

Librerías nativas

Esta segunda capa de la pila que forma Android también tiene un conjunto de librerías escritas en C (ISO, 2011) o C++ (ISO, 2014) usadas por varios componentes de Android. Estas librerías son de gran utilidad, muchas de ellas de código abierto y son expuestas para su uso por las apps a través de la API del framework de JAVA que proporciona la plataforma de Android. Algunas de estas librerías son:

- **System C**. Módulo de C estándar adaptado a dispositivos embebidos en Linux
- **WebKit**. Que da soporte al navegador de Android y a la vista WebView
- **FreeType**. Librería de fuentes del sistema en bitmap
- **OpenCore**. Soporte de códecs de reproducción y grabación de audio y video.

- **SGL**. Motor de gráficos 2D
- **SQLite**. Motor ligero de base de datos
- **Surfacemanager**. Maneja el acceso al subsistema de representación 2D y 3D.
- **OpenGLES**. Acceso a la librería de aceleración hardware para gráficos 3D.
- **SSL**. Proporciona servicio de encriptación.

Entorno de aplicación

También conocido como Framework de la Java API. Esta capa proporciona un entorno de desarrollo basado en un lenguaje bien conocido como es JAVA y aunque no incluye todo el JRE de JAVA, sí que se encuentra, en el SDK de Android, una gran parte de él.

Esta capa proporciona también acceso simplificado a gran parte de componentes del sistema, entre los que destacan:

- **Sistema de Vistas.** Conjunto de vistas proporcionadas por el sistema
- **Manejador de notificaciones.** Para que cualquier aplicación pueda mostrar una notificación en cualquier momento que o necesite
- **Manejador de recursos.** Da acceso a recursos que no tienen código, como pueden ser imágenes, Strings, audios, videos o archivos de diseño
- **Content Providers.** Permite acceder a datos de otras apps y compartir datos con las demás
- **Manejador de actividades.** Administra el ciclo de vida de las aplicaciones y además controla la pila de retroceso del botón volver de Android
- **Manejador de ventanas.** Gestiona las ventanas de la aplicación

Aplicaciones

Este último nivel es en el que se encuentran todas las aplicaciones instaladas en el sistema, ya sean las propias del sistema o de terceros instaladas por el usuario. Todas han de ejecutarse sobre la máquina virtual, ya sea Dalvik o ART y pueden estar escritas o en JAVA a través del SDK o en código nativo C/C++ a través del NDK.

2.2 Xgestevo

2.2.1 ¿Qué es Xgestevo?

Xgestevo es un software de ERP (Sistema de planificación de recursos empresariales o Enterprise resource planning), que nace en Cartagena en 1998 de mano de Software de Gestión XGEST, S.L. con la idea de cubrir la mayoría de las necesidades de un PYME.

Es un software en continuo desarrollo, adaptándose a los cambios legislativos y normativos, como los ocurridos con SEPA, IVA, modelo 347 o Inversión de Sujeto Pasivo y condicionamiento de revendedor.

La distribución es realizada por profesionales de la informática (autónomos, tiendas de informática y mayoristas) que se encargar de instalar y dar soporte a sus clientes. Queda así libre el equipo de desarrollo para crear, actualizar y corregir la aplicación y posteriormente formar o reciclar a dichos profesionales en todas las novedades que pueda tener el sistema.

Es un software de pago mensual por suscripción, cuyo licenciamiento se hace por terminales que usan el sistema y servidores en los que está instalado. En su módulo base se incluyen las siguientes características:

- Gestión de almacén
- Gestión del proceso de compras
- Gestión del proceso de ventas
- Proceso centralizado de facturación
- Creación y administración de facturas electrónicas con firma digital, incluyendo formato oficial Facturae
- Gestión de cobro a clientes y su control del riesgo
- Gestión del pago a proveedores
- Modo TPV
- Gestión de Servicios Técnicos (SAT)
- Gestión de devolución de material de clientes y proveedores (RMA)
- Integración de producto, ya sean montados o fabricados
- Trazabilidad de productos por control de lotes
- Seguimiento por número de serie de productos de compra y venta
- Reserva de pedidos de cliente
- Seguimiento de incidencias en los clientes y proveedores
- Generación de informes de previsión de compras y posibilidad de generar el pedido al proveedor automáticamente.
- Control logístico de etiquetado y seguimiento de bultos.
- Generación de remesas SEPA a través de la gestión de cobros a clientes
- CRM con posibilidad de comunicación automática por mail y gestión de puntos
- Contabilidad integrada con todos los elementos de gestión, según el Plan General Contable español

- Contabilidad analítica de costes

Como módulos adicionales y con licenciamiento independiente, tiene los siguientes módulos:

- Gestión documental, totalmente integrada en la base de datos, sin necesidad de carpetas compartidas y con posibilidad de volcado directo del documento, desde un dispositivo de almacenamiento o escáner, al sistema de gestión
- Gestión avanzada de partes de trabajo.
- Cubos OLAP – Generación de informes manejando gran cantidad de datos.
- Tienda Web integrada con el sistema, con posibilidad de comportamiento en línea o con sincronizaciones periódicas
- Dispositivos móviles.
    - Sistema de Preventa, con sincronización offline
    - Módulo de control de almacén
    - Gestión de ordenes de trabajo de los técnicos.

Algunas de las características más destacables que se encuentran en Xgestevo son las siguientes:

- Arquitectura cliente/servidor basada en BBDD MySQL/MariaDB con posibilidad de alojamiento local o remoto
- Acceso remoto a la aplicación ya sea con el propio cliente Xgestevo o con escritorio remoto.
- Múltiples empresas
- Múltiples almacenes
- Múltiples series de facturación
- Todos los ejercicios contables en línea desde el principio del uso de la aplicación
- Impresos totalmente diseñables.
- Control de acceso definido por usuario o grupo
- Generación de ficheros PDF y Excel para cualquier documento o informe generado. Se incluye la posibilidad de mandar directamente por correo electrónico tras su creación.
- Permite tener multitud de vistas abiertas al mismo tiempo (MDI)
- Posibilidad de gestionar estándares de calidad ISO
- Creación automática de remesas SEPA B2b y CORE
- Facturación periódica recursiva
- Gestión de la contabilidad interna.

- Puede gestionar varios diarios contables independientes, máximo diez

2.2.2 Breve historia

Antes de 1998, año en que empieza el proyecto Xgest, la empresa encargada de su diseño y mantenimiento, tenía otros programas de gestión comercial y contable programados en Xbase: DBase III (dBase, 2017), Clipper (Spence, R., 1993) y Foxpro (Microsoft Corporation, 1994) para MS-DOS.

A principios de 1998, tomando de base los programas de gestión y contabilidad de los que ya disponían, comienzan una nueva implementación en un entorno visual para Windows. El lenguaje elegido para dicha implementación sería Visual Foxpro 6 (Microsoft Corporation, 1998), pues los anteriores desarrollos se habían realizado en Foxpro 2.6 para DOS.

El cambio fue bastante complicado, ya que la programación basada en objetos y eventos en formularios para Windows es muy distinta de la que se usaba en MS-DOS.

Se pretendía además usar una base de datos cliente/servidor para evitar los problemas que daba el sistema de las anteriores versiones, que era la competición directa de ficheros a nivel de bloqueo para su uso en múltiples puestos. En Windows no había bases de datos gratuitas aún, aunque casualmente a los pocos días, el 8 de enero, se publicó la primera Release para Windows como se puede observar en MySQL News (2008), por lo que se probó PostgeSQL y MySQL en Linux. Por aquel entonces todas las pruebas demostraron, que para el proyecto que se pretendía crear, MySQL era muchísimo más rápida y por lo tanto fue la elegida.

Se desarrolló un framework de objetos visuales para introducción de datos vasados en los objetos base de Visual Foxpro que actualizan automáticamente los datos de MySQL simplemente usando ciertos parámetros en los formularios para evitar tener que hacer una consulta SQL cada vez que se cambia un campo.

Finalmente, el 1 de enero de 1999 se pone en marcha la primera instalación en el mayorista de informática Depau Sistemas. Esta primera instalación se realizó en clientes con Windows 95 y el servidor de MySQL en un Linux RedHat.

Aunque basado en estructuras anteriores, el desarrollo de Xgest se hizo teniendo muy en cuenta el efecto 2000, al que solo un año después de su puesta en funcionamiento debería de enfrentarse. Por ello todas las fechas con las que trabaja son con años de cuatro dígitos.

A finales del año 2000 sale la primera versión de XWeb, una web/extranet básica, que los usuarios de Xgest podían usar para dar servicio a sus clientes. Inicialmente se hace bajo IIS con FoxISAPI (Visual Studio .NET, 2003), pero dada la dificultad en el desarrollo, instalación y depuración, pronto se cambió a una versión en PHP bajo Apache obteniendo muchos mejores resultados, tanto en el desarrollo como de cara a los clientes.

Otro de los primeros grandes retos a los que se enfrentó fue al cambio del EURO. Era un problema algo más complejo que el del efecto 2000, pero al haberse previsto y saberse el cambio fijado con tiempo, se prepararon unos procedimientos de conversión, teniendo en cuenta redondeos y demás factores. El 1 de enero de 2002 se realizó la transición sin problemas.

A principios de 2007 alguna de las empresas que usaban Xgest ya eran demasiado grandes para la última versión que había de Xgest v.7. La capacidad de almacenamiento, la limitación en el número de clientes y el límite en las numeraciones en los documentos, hicieron que algunas de las más grandes tuvieran problemas para terminar el año 2006, por lo que se hacía necesario un cambio profundo.

En Julio de 2007 sale la primera beta del nuevo sistema Xgestevo, cuyas principales características fueron las siguientes:

- Cambio de la base de datos MySQL a IBM/DB2 V9 (IBM, 2006). Versión libre de la base de datos profesional de IBM con capacidades de hasta 4GB de RAM, 2 procesadores y almacenamiento ilimitado.
- Integración en un único programada el software de gestión Xgest, el de contabilidad XCONTA y XTPV, así como de otras utilidades que hasta ese momento estaban en programas externos.
- Ampliación de los códigos de cliente a 6 dígitos hasta 999999 y de las cuentas contables a 10 dígitos para permitir dichos códigos (hasta el momento eran los clientes 5 dígitos y las cuentas contables 9)
- Ampliación a 6 dígitos el número de documento anual y a 4 el número de serie, cuando hasta el momento eran 5 dígitos el número de documento y 2 el de serie. Quedando el formato de numeración de documentos de la siguiente manera:

XXXX YYYY ZZZZZZ

Siendo x el número de serie, y el año, z el número de documento, por tanto:

0001 2017 000001

Sería el primer documento de la serie 1 del año 2017

- Gestión de usuarios totalmente integrada con los menús de la aplicación y con posibilidad de usar grupos de usuarios a la hora de asignar privilegios.
- Estructura de menús dinámicas, solo muestra las opciones que tiene disponible cada usuario
- Compatibilidad mejorada con Terminal Server

El 1 de enero de 2008 entra en funcionamiento la primera versión de Xgestevo (también conocida como la versión 8.0 de Xgest), aprovechando la entrada en vigor del nuevo Plan General Contable, que sería ese mismo día, y para el cual estaba esta versión totalmente adaptado.

Con la compra de MySQL en 2008 por Sun Microsystems (EFE, 2008) y un año después, en 2009, la compra de Sun por Oracle (EP, 2009), hace que la empresa sienta miedo por el futuro de la base de datos de la mitad de sus instalaciones que aún están en MySQL y Xgest v.7 por lo que se implementa una rutina de conversión en la rutina principal por la que pasan todas las instrucciones que se envían a la base de datos por OBDC. Con estas rutinas de conversión consiguen que Xgest v.7 pueda usarse, además de con MySQL, en PostgreSQL y con IBM DB2 y Xgestevo amplía su uso también a MySQL y PostgreSQL, además de con IBM DB2. El cliente es libre de elegir, aunque cada versión tiene su base de datos por defecto, y todas las nuevas instalaciones se hacen en IBM DB2.

DB2 resulto ser una base de datos demasiado pesada para los mayores clientes de Xgestevo y demasiado complicada de mantener en las empresas pequeñas. Además, tenía graves problemas a la hora de recuperar una tabla dañada, lo cual hizo bajar sensiblemente la fiabilidad de la aplicación. Finalmente, en 2011 se toma la decisión de migrar todas las bases de datos y eliminar DB2 completamente. Tras su eliminación, se incorpora MariaDB como alternativa a MySQL y PostgreSQL.

MariaDB (MariaDB, 2016), un fork libre y gratuito de MySQL, pasa a ser la base de datos por defecto de Xgestevo.

Actualmente da servicio a más de 800 empresas en toda España y con más de 3000 terminales de trabajo. Siendo alguno de sus mayores casos de éxito, empresas como Depau Sistemas con más de 100 terminales funcionando simultáneamente.

2.2.3 Versiones de Xgestevo

La diferencia entre versiones de Xgest no era muy profunda, se iban liberando versiones conforme se iban añadiendo nuevas funcionalidades, como se sigue haciendo en

la actualidad (última versión 32.05 del 17 de abril de 2017). Se sube de versión normalmente cuando se van ampliando campos en la base de datos.

Las versiones más significativas y a las que merece la pena mencionar serían:

Xgest V1. La versión original que se pone en producción el 1 de enero de 1999.

Xgest V5. Salió en 2001, pero en 2003 se convirtió en una versión gratuita totalmente operativa y sin caducidad. Hecha para animar a las pequeñas empresas a que lo probaran y pasaran a pagar soporte cuando la usaran más a fondo. A finales de 2016 aún sigue alguna instalación funcionando con esta versión.

Xgest V7. Se lanza en 2003 y también fue llamada Xgest7. Fue la última versión que aún mantenía la estructura separada de un software de gestión comercial (Xgest) y la contabilidad (Xconta). Fue la que durante más tiempo ha mantenido el número de versión (V7), pues se mantuvo hasta la aparición de Xgestevo.

Xgestevo. A partir de enero de 2008, todos los módulos que componen la familia Xgest se unen en Xgestevo. Sería la versión 8 del sistema, pero a partir de este momento se mantiene el nombre y las versiones no implican grandes cambios como ocurría en versiones anteriores.

2.2.4 Estructura de Xgestevo

Xgestevo se basa en una estructura cliente/servidor. En el cual existe un servidor en MySQL o MariaDB, ambos en versiones totalmente compatibles. Y múltiples clientes instalados en equipos con Windows que conectan con este servidor.

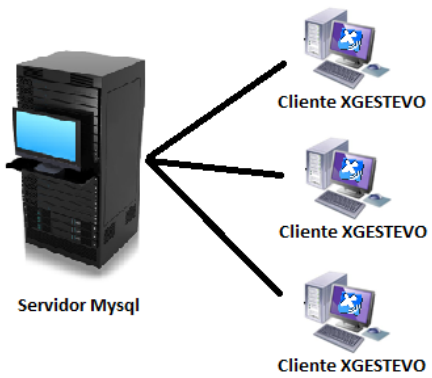

Ilustración 4. Estructura cliente servidor

La base de datos se compone de un total de 263 tablas en su instalación básica para una única empresa, de los cuales 27 son para contabilidad, 223 para gestión y 13 para históricos. Hay que recordar que Xgestevo es multiempresa, por lo que puede gestionar

hasta un total de 999 empresas con una única licencia, muchas de las tablas se tendrán que volver a generar por cada empresa que se cree.

De las 27 tablas de contabilidad 4 son de propósito general y comunes a todas las empresas y en las que se codifican tipos de cuentas, tipos de cargos a clientes y proveedores y los modelos del Plan General Contable.

Las otras 23 tablas se crean por cada empresa que se tenga y que a su vez tienen otra subdivisión, pues 9 de estas tablas son de propósito general a cada empresa, teniendo los datos contables de clientes, proveedores y de la misma empresa y las otras 14 tablas se generan con cada año contable, por lo que cada año hay 14 tablas más. Estas últimas tablas contienen los movimientos, asientos, remesas, IVA y demás datos necesarios para llevar el año contable de clientes y proveedores.

De las 223 tablas de gestión, 28 son de índole general para toda la instalación y contienen los datos de usuarios, permisos, listados de empresas, agencias de transporte, configuraciones web, configuraciones de correos y demás datos de uso común y para la administración de la instalación.

Las otras 195 tablas contienen toda la información de ofertas, pedidos, albaranes, facturas de clientes y proveedores, definiciones de formas de pago, artículos, servicio técnico, RMA, ofertas y pedidos de web y TPV, fotografías de artículos, PDF o XML de facturas firmadas digitalmente y muchas otras más que son necesarias para el correcto funcionamiento de la aplicación.

Finalmente tenemos 13 tablas que son históricos de las tablas más grandes del sistema de gestión, que son los pedidos, ofertas, albaranes y facturas de clientes y proveedores. Algunas de estas tablas llegan a tener millones de registros anuales, por lo que en su momento se implementó una rutina que archivara estos datos una vez pasados los 5 años que exige la ley que se mantengan accesibles en todo momento. Siguen siendo accesibles, pero a través de consultas en el histórico y no en las rutinas generales del programa.

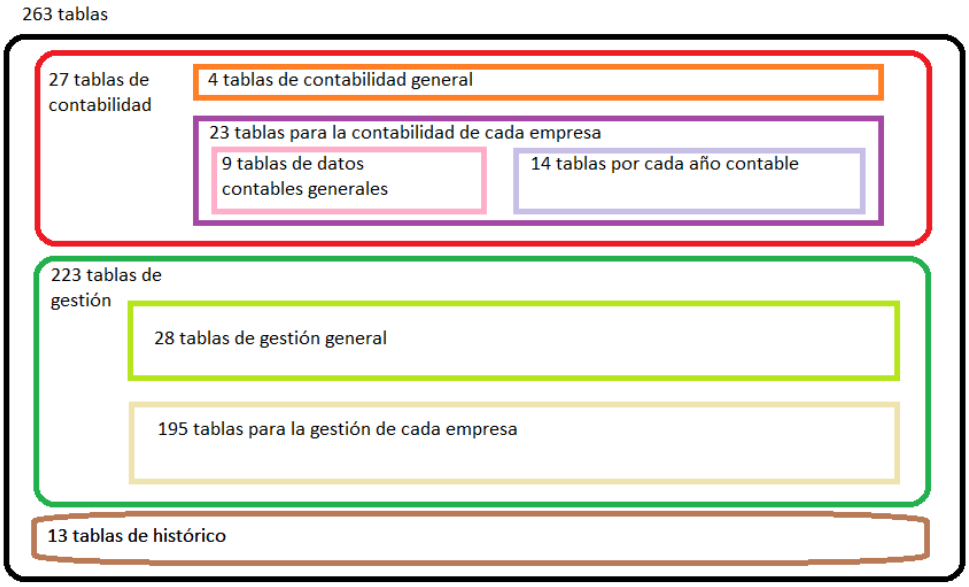

Ilustración 5. Estructuración de las tablas de Xgestevo

La aplicación de escritorio, como ya se ha dicho antes, está basada en Windows, pero se le ha dotado de una alta funcionalidad en Terminal Server para ser accedido tanto de manera remota por otro sistema Windows o por cualquier otro sistema, ya sea Linux, MacOS, Android o IOS. Sus funcionalidades en el módulo básico fueron definidas en el apartado de introducción de Xgestevo, por lo que se verán a continuación sus módulos opcionales.

Tienda Web y Extranet

Este módulo ha sido desarrollado en PHP y funciona sobre un servidor Apache 2, usando un framework bastante flexible como es Laravel (Laravel, 2017).

La web es totalmente parametrizable desde el propio Xgestevo, hasta tal punto que cualquier modificación en artículos, clientes, estado de pedidos, ordenes de trabajo o cualquier otra funcionalidad que ofrece el sistema y que se quiera que los clientes vean, es automáticamente sincronizada con la web y vista por los clientes.

Gestión documental

Otra opción que se incluyó fue la gestión documental. En principio basada en un sistema de archivo de certificados de calidad de materiales para cumplir las normativas ISO, se amplió para que pudiera adjuntarse cualquier tipo de archivo informático (documento, hoja de cálculo, imagen, PDF, etc.) vinculado a cualquier documento de la aplicación (factura, asiento contable, cliente, proveedor, pedido, etc.). Permite también escaneo directo

desde la aplicación directamente desde dispositivos compatibles TWAIN para mayor agilidad.

En un principio, para evitar problemas de tamaño estos archivos se guardaban en un directorio compartido, compartimentado en directorios con 1.000 archivos cada uno para evitar los problemas que daban los antiguos Windows con directorios con muchas entradas de directorio. Este sistema tenía el problema de que solo podían acceder a los archivos los ordenadores en la red local del servidor, no los conectados de forma remota. Más adelante se cambió a su formato actual, que almacena esos archivos directamente en campos binarios de tablas de la misma base de datos. Con esto se solucionan todos los problemas de acceso remoto, aunque lógicamente la base de datos tiende a hacerse bastante grande.

Inteligencia de negocio

En 2007 se adquiere e incorpora la licencia libre de royalties de Contour Cube, un control ActiveX que permite crear cubos OLAP donde se cargan volúmenes de datos enormes y permite manejar esos datos de una forma muy ágil cambiándolos entre el eje X e Y añadiendo o quitando, resumiendo o detallando todo hasta que se obtiene el resultado deseado, que puede ser exportado a una hoja de cálculo o generar gráficos. Como opción dentro de la aplicación, pueden generarse cubos de distintos movimientos de compras, ventas, facturas, etc.

Movilidad

Este módulo es uno de los elementos complicados debido a la cantidad y rápida evolución de los distintos dispositivos y sistemas disponibles a lo largo del tiempo.

En principio se desarrolla el módulo como un sistema totalmente offline programado en Windows Mobile/CE, bastante básico y con sincronización a través de conexión directa de los dispositivos a un PC con el cliente de Xgestevo.

Después se desarrollaron módulos de gestión de almacén online y gestión de prevente offline con sincronización por medio de FTP, todo ello con Windows Mobile 5/6.

Microsoft dejó obsoleto este sistema con la aparición en 2010 de Windows Phone 7 como sustituto de Windows Mobile 6, ya que ambos sistemas no eran compatibles. Los dispositivos no se podían actualizar y la compañía decidió mantener el sistema con Windows Mobile hasta que finalmente el mercado mostrara que sistema terminaría imponiéndose: Windows Phone, IOS o el recién aparecido Android.

Caso real de instalación con módulos opcionales

En la siguiente figura se muestra el esquema de un caso real de instalación de Xgestevo, en el que se usan varias de sus funcionalidades.

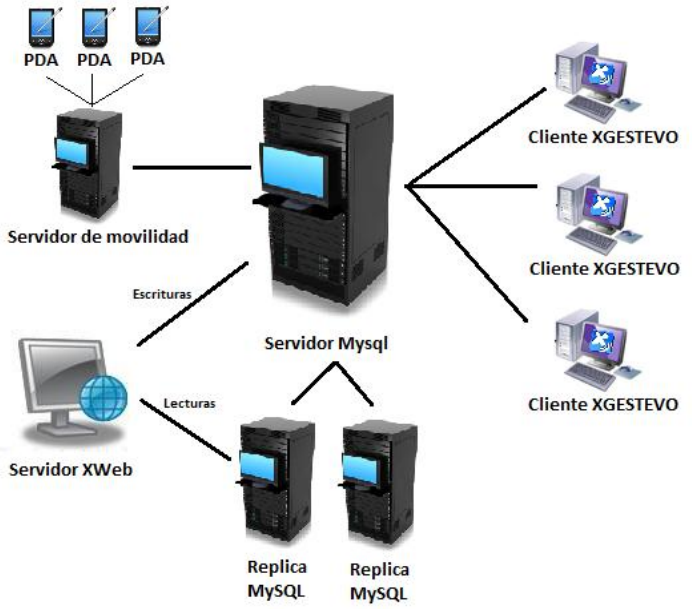

Ilustración 6. Ejemplo de instalación en Depau Sistemas

La figura muestra que la instalación hace uso del módulo básico de servidor y clientes, pero además usa el servidor de movilidad junto con terminales con Windows Mobile 6.5.

Además, dispone de varias réplicas del servidor principal, útiles como copia de seguridad y además son usados para otros dos módulos opcionales del sistema. Xgestevo está preparado para poder usar un servidor de réplica para el uso de la inteligencia de negocio, y por tanto poder generar los Cubos OLAP con grandes cantidades de datos sin molestar al resto de usuarios del servidor principal. El segundo módulo opcional que hace uso de los servidores de réplica es la web, que, para usos con muchas visitas, se puede configurar para que las lecturas producidas por la navegación normal sean realizadas en estos servidores esclavos (ya sean en las instalaciones de la empresa o en la nube), mientras que las escrituras en la base de datos, se derivan al servidor principal o maestro.

3. El almacén según Xgestevo

Xgestevo gestiona todas las operaciones del almacén con una sección independiente de control de artículos, que por norma general administra el personal de catalogación, o responsable del alta y mantenimiento de los artículos.

Este módulo de artículos, dentro del Xgestevo, sirve de intermediario entre los otros dos módulos principales, el de gestión de proveedores y el de gestión de clientes.

Todas las tablas de las que se hablará a continuación aparecerán desglosadas y explicados sus principales campos en el Anexo 1

3.1 Control de Stock en Xgestevo, sus tablas y operaciones

Xgestevo lleva el control completo sobre los artículos desde una única ventana. Todos los datos pertenecientes a esta ventana provienen, sobre todo, de la tabla fcartxxx, que es la que almacena por cada artículo que haya en la base de datos, toda la información de interés relacionada con las características del articulo y su catalogación, además del stock general del artículo, o lo que es lo mismo, la suma de stock de todos los almacenes.

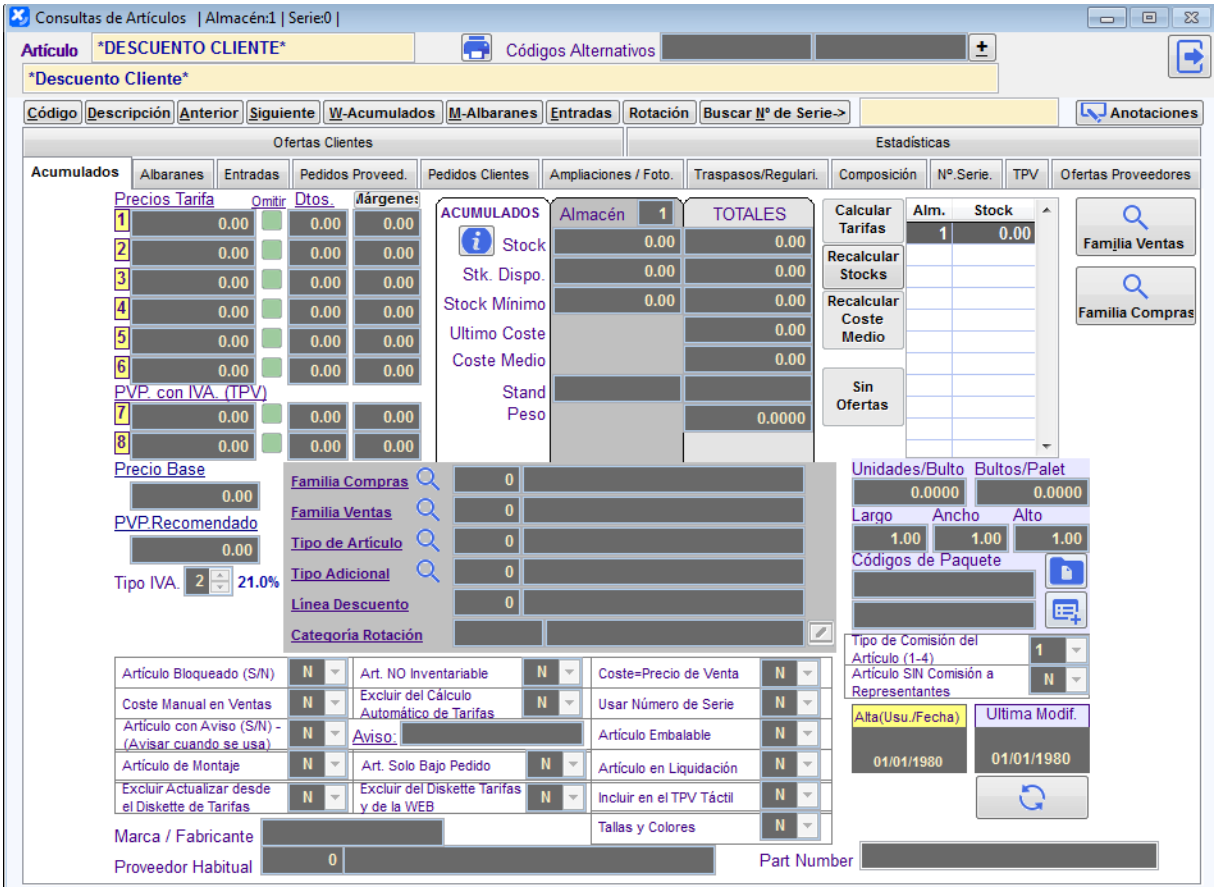

Ilustración 7. Ventana de consulta de artículos de Xgestevo

Otra de las tablas de las que saca información esta ventana y que también es de una gran importancia, será la tabla fcstkxxx que almacena por cada artículo y cada almacén que tengamos en la empresa, el stock actual.

Es sobre estas dos tablas sobre las que el resto de procesos de Xgestevo actúan cuando se realizan operaciones de clientes como son las ofertas, pedidos, albaranes y facturas o las operaciones de proveedor, también ofertas, pedidos, entradas y facturas. Todas ellas apoyándose en el stock que estas dos tablas proporcionan según la necesidad de la operación que realicemos.

Hay más tablas que apoyan al mantenimiento de artículos, pero solo la tabla fcfosxxx resulta de interés verdadero, pues almacena el contenido multimedia relacionado con el artículo, principalmente fotos.

3.2 Proveedores en Xgestevo, sus tablas y operaciones

Proveedores tiene una sección entera en Xgestevo con multitud de operaciones, como podemos ver en la imagen que acompaña a este texto.

Todas son operaciones necesarias relacionadas con el proceso de entrada de material en el almacén, manufactura de productos tomando como base material del almacén (Integración) y principalmente, maneja la relación de la empresa con los distintos proveedores.

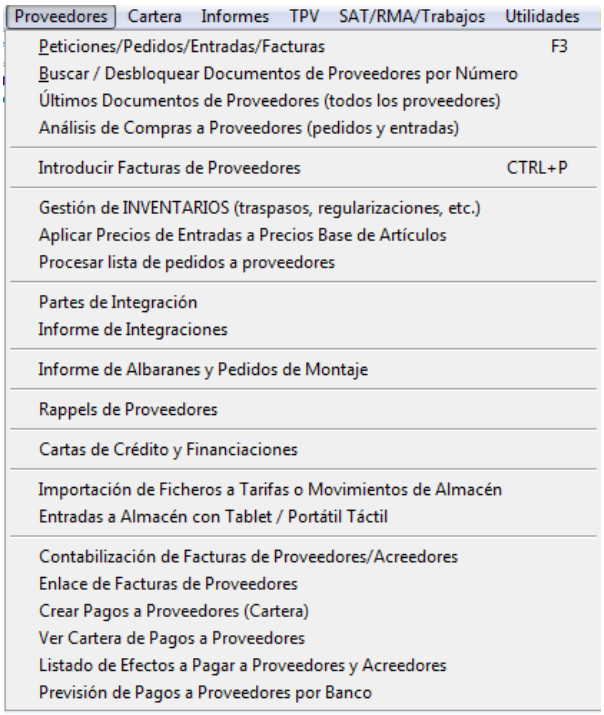

Ilustración 8. Menú de proveedores de Xgestevo

La principal de las secciones, en este menú, es la primera de todas: Peticiones/Pedidos/Entrada/Facturas.

En esta opción es donde se encuentran todas las operaciones, relacionadas con el material y sus operaciones con el almacén, que se realizan con los proveedores. En ella se pueden crear ofertas que se le solicitan al proveedor. Una vez que estas ofertas son aceptadas por el personal de compras, se podrán pasar a pedidos, que a su vez se convertirán en entradas una vez que lleguen al almacén y que finalmente serán facturadas para realizar el pago por el material servido.

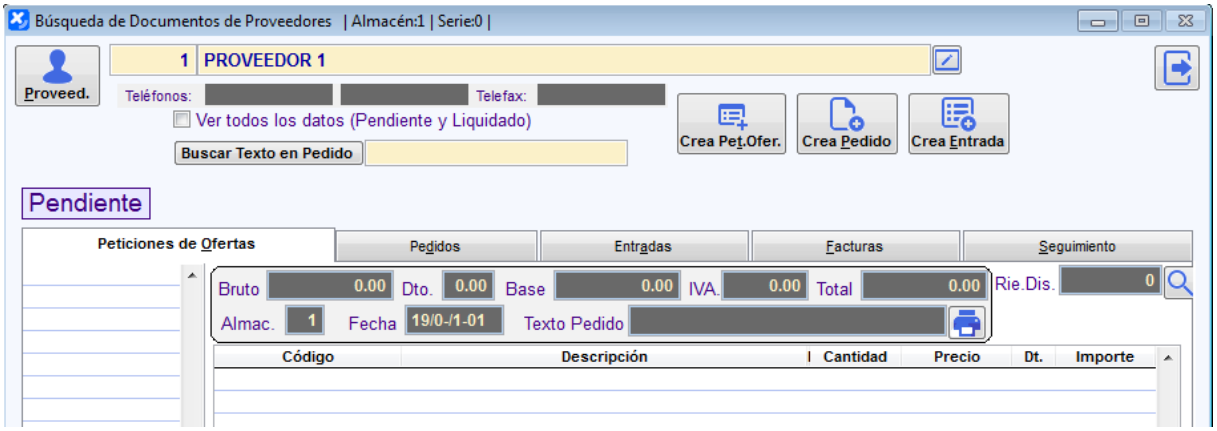

Ilustración 9. Ventana de proveedores de Xgestevo

La principal tabla relacionada con los proveedores sería, como es evidente, aquella que contiene todos los datos relacionados con este, como pueden ser sus datos fiscales y de contacto. Todos ellos se guardan en la tabla fcproxxx.

A continuación, puede verse la primera aparición de una de las principales características de todas las operaciones relacionadas con el movimiento de almacenes, y es que toda operación de entrada, salida o reserva está sustentada por dos tablas: una tabla de cabecera que contendrá los datos generales de la operación, como son el destinatario, el almacén en el que se realiza, la forma de pago los datos económicos relacionados con la operación. La otra tabla seria la que contiene todas las líneas relacionadas con la operación, en ellas se dice sobre que artículo se opera, cantidad, condiciones económicas especiales y el estado en el que se encuentra actualmente en la operación.

La primera de estas operaciones es la de ofertas de proveedor que se apoya en la tabla fccopxxx para almacenar los datos generales de la oferta que se le ha solicitado al proveedor y la tabla fclopxxx que almacenará a su vez una línea por cada artículo que se la haya solicitado oferta al proveedor.

La segunda operación será la de pedido a proveedor. Esta es una operación curiosa en cuanto al manejo de Xgestevo, pues el sistema la considera como un estado avanzado de una oferta y por tanto usa las mismas tablas que para estas, fccopxxx para las cabeceras y fclopxxx para las líneas, solo que usa dos numeraciones y fechas distintas para cada estado y un campo que dice en cuál de los estados se encuentra, si en oferta o en pedido. También incluye campos para indicar qué cantidad del total ha pasado a entrada y cual queda pendiente.

La tercera operación es la de entrada de proveedor y en esta ocasión sí que actúa sobre el stock, pues en el momento que haya una línea en la tabla fcliexxx, que es la que registra una línea por cada artículo que entra en el almacén, se producirá en Xgestevo un recálcalo sobre las tablas fcartxxx y fcstkxxx. Como en los casos anteriores, existe también una tabla para la cabecera de las entradas, que es fccbexxx.

La última operación sobre proveedores es la de facturación. Esta operación no afecta al almacén, solo afecta a nivel contable, generando los respectivos asientos y enlaces en las remesas de contabilidad, necesarios para el control de pagos a proveedores. La tabla implicada en esta operación será fcfprxxx que contendrá la información del importe de la factura y datos de enlace con tablas de contabilidad, como cuenta contable a la que se asigna y el número de asiento.

3.3 Clientes en Xgestevo, sus tablas y operaciones

Al igual que ocurre con proveedores, el manejo de los clientes también tiene una sección completa en Xgestevo, incluso más amplia que la anterior.

En esta sección se encuentran todas las operaciones que el sistema puede realizar de cara a operaciones de preventa y venta a cliente. Incluye acceso simplificado a la información financiera de los clientes, estado de la logística relacionada con las ventas, sistema de facturación y gestión de documentación con los clientes.

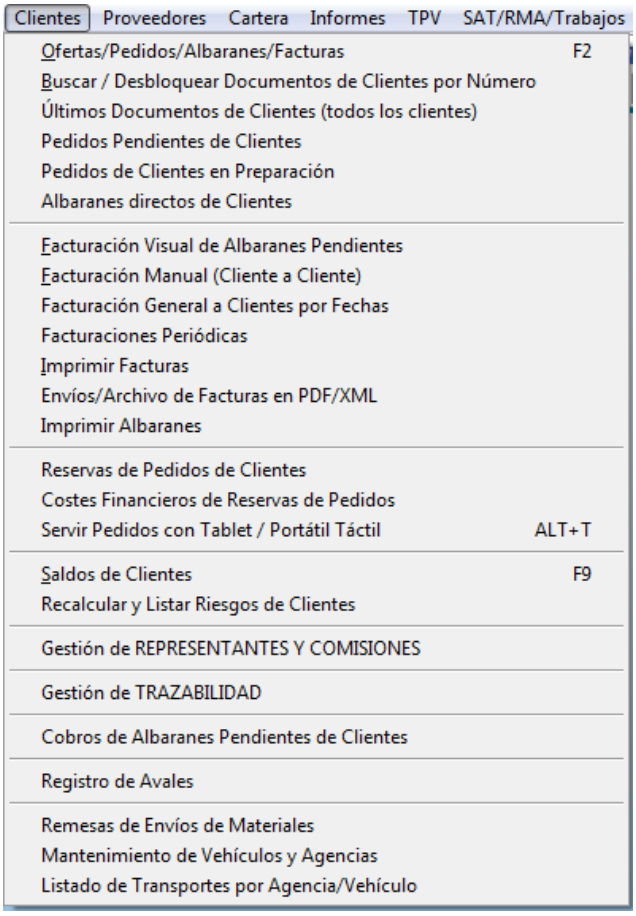

Ilustración 10. Menú de clientes en Xgestevo

Al igual que con proveedores, la sección que controla las operaciones directas de los clientes con el almacén es la que más uso tiene y la que más interés tiene en este proyecto. La sección en cuestión se llama Ofertas/Pedidos/Albaranes/Facturas. En ella se encuentran todas las operaciones que el personal de ventas puede realizar con un cliente y que afecta directamente sobre el estado del almacén. Es una sección muy parecida a la de proveedores y las operaciones muy similares, pero con relación a los clientes. Se pueden hacer ofertas o presupuestos sobre material que nos solicita el cliente; este material una vez que se confirma, por parte del cliente, pasará a pedido y se reservará en el almacén en caso de haber stock, por lo que dejará de estar disponible para otros clientes; la principal operación de modificación de stock en los clientes será el albarán, que indica que el material ha salido del almacén y por tanto descontará la cantidad necesaria del stock; por último las facturas con las cuales se genera el pago que el cliente debe hacer por los productos servidos.

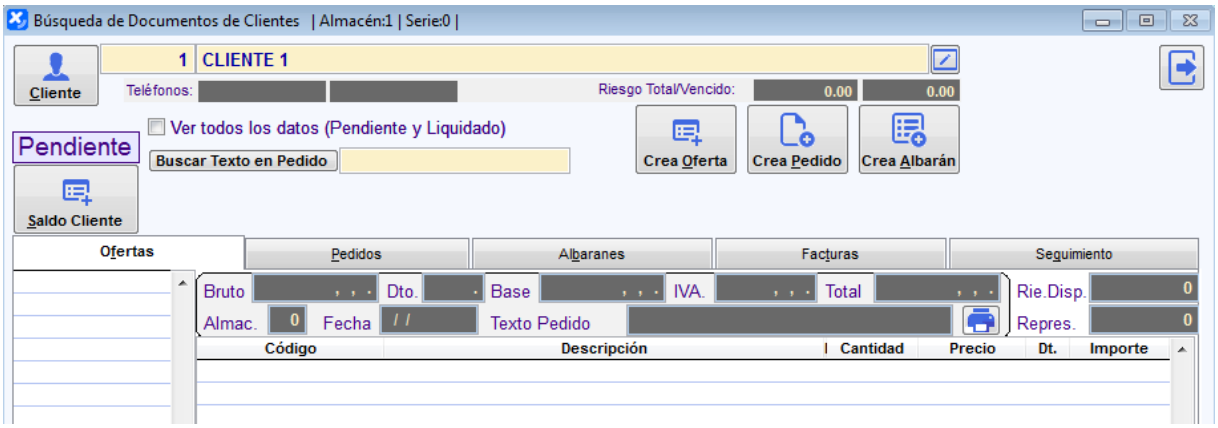

Ilustración 11. Ventana de clientes en Xgestevo

La principal tabla relacionada con los clientes en Xgestevo es fcclixxx, en la cual se guardan los datos personales, fiscales y financieros de los clientes.

Ocurre lo mismo que con proveedores, casi todas las operaciones con los clientes también constan de dos tablas por operación.

La primera de las operaciones será la de ofertas, que se gestiona con la tabla fccocxxx para las cabeceras. En ella se guardan los datos generales del presupuesto entregado al cliente, el importe total y los datos del cliente. La otra tabla relacionada será la que se crea por cada uno de los artículos incluidos en la oferta. La tabla se llama fclocxxx y almacén una línea por artículo que ofertemos.

La siguiente operación es la de pedidos de cliente y al igual que ocurre con los proveedores, otra vez, se gestiona con las mismas tablas que las ofertas. Están también los mismos campos que indican el número de oferta y de pedido, sus respectivas fechas y si está en un estado o en otro. También incluye los campos que controlan la cantidad pendiente de server y la ya servida, pero además hay campos extras para controlar si el material está reservado o no.

La operación de salida de material del almacén, o la creación del albarán, es como en las entradas de proveedor, la que realiza el verdadero movimiento de stock en el almacén. Está controlada por la tabla fccbaxxx para las cabeceras y por la tabla fcliaxxx para las líneas del albarán.

Por último, la operación de facturación, que es la encargada de iniciar el proceso financiero con el cliente, al igual que pasaba con los proveedores. Las tablas encargadas de esta gestión serán dos en este caso, la tabla fcfacxxx que almacena todos los datos de la factura en si al igual que pasaba con la de proveedor, pero está apoyada por otra tabla

llamada fcpdfxxx que almacena una copia en PDF, o en XML de la factura, incluyendo si fuera necesario una firma digital para asegurar que no sea modificada una vez creada.

Debido a que las operaciones con los clientes pueden llevar un sistema de envíos asociado a cada operación, existen otras tablas que almacenan otros datos del cliente y que pueden ser de interés. Dos de ellas son las que más interés tienen, la tabla de centros fccenxxx que almacena distintos puntos fijos de entrega asociados al cliente y la tabla fcdrexxx que se usará para asociar ofertas, pedidos y albaranes con una dirección de envió específica para la que el cliente solicita el envió del material en formato dropshipping.

3.4 Software de control de almacén para Xgestevo. Alternativas y conclusiones

El objetivo de este proyecto es la creación de un módulo de control de almacén que pueda llegar a sustituir al actual software que se usa con Xgestevo. Este software se denomina Regumobile y en esta sección se verá su utilidad, las alternativas que hay y finalmente, la necesidad de un software nuevo.

Durante muchos años, todas las operaciones que se hacen en el almacén en Xgestevo, se han tenido que hacer manualmente, teniendo que ver en un ordenador de sobremesa toda la información necesaria.

Todo el material tenía que ser arrastrado hacia esos terminales fijos para poder introducir manualmente los códigos de los productos y poder de esta manera realizar las operaciones típicas del almacén: entrar en la ficha del articulo para poder consultar el stock, o poder modificar cualquiera de sus características; poder entrar en la ficha de un proveedor y hacer directamente una entrada del material que iba acercando el personal responsable de revisar que el material llegara correctamente desde el proveedor; o por otra parte, poder generar en la ficha de los clientes, un albarán directo con todo el material que previamente se ha ido buscando en el almacén.

Con la bajada de los precios en los lectores manuales de códigos de barras, se logró un avance al no tener que escribir manualmente los códigos de cada artículo, con lo que se aceleraba de manera considerable cualquier operación en el almacén.

Por otro lado, se crearon rutinas que relacionaban los pedidos de proveedor con las entradas y los pedidos de clientes con los albaranes. Gracias a esto y con los lectores de códigos de barras, se podía servir el material desde un pedido y generar la correspondiente entrada o albarán, solo con el material que realmente era necesario y sin posibilidad de

errores. Esto dio lugar a la mejora de la eficiencia en las operaciones, pues se cometían muchos menos fallos al verificar articulo a articulo con lo que se había pedido.

Pero aún existía un gran problema, la gente del almacén tenía que consultar permanentemente las pantallas de los terminales fijos o imprimir listados de material para localizar tanto el material de entrada como el de salida. Lo cual ocasionaba una enorme pérdida de tiempo y muchos desplazamientos del personal, que finalmente derivaba también en fatiga.

Surge entonces la necesidad de crear terminales móviles para intentar paliar dicha situación. El primer intento y más obvio, se realiza en 2005 con la estandarización de los portátiles y las redes wifi en el mercado. La solución era muy simple, instalar un portátil y un lector de códigos de barras en un carro y usar el propio software Xgestevo para realizar las mismas operaciones que se hacían en los puestos fijos.

Fue una solución válida y utilizada por varios clientes con medianos y grandes almacenes, pero presentaba varios problemas: el caro con el portátil era demasiado aparatoso para moverse por pasillos; las baterías duraban como mucho 1,5 o 2 horas, por lo que cada usuario necesitaba cambiar entre 3 y 5 veces la batería en su jornada de trabajo, con el riesgo de no darse cuenta y que se apagara el portátil, perdiendo todo el trabajo realizado. También estaba el problema de usar un software demasiado complejo como es Xgestevo para hacer operaciones muy simples, era necesario simplificar las operaciones y dejar solo las mínimas opciones necesarias al personal de almacén para hacer su trabajo y no que interfiriera con operaciones de compras, ventas o mantenimiento de artículos. Por último, se había llegado ya a 2008, España estaba en crisis y las empresas necesitaban optimizar tiempos, recursos y costes, por lo que todos estos anteriores problemas con los que dese 2005 se había podido convivir, ahora era fundamental resolverlos

Con estas últimas premisas establecidas, surge la necesidad de realizar un módulo específico para el control del almacén, que fuera ligero, con las opciones imprescindibles y que pudiera instalarse en algún dispositivo que fuera fácil de manejar y que tuviera suficiente autonomía. Surge en 2009 el módulo Regumobile.

3.4.1 Regumobile

Regumobile fue la solución que diseñaron los desarrolladores de Xgestevo. Un software desarrollado específicamente para usarse con Xgestevo en terminales móviles con Windows CE OS en su versión 5, con instalaciones en Mobile 5, Mobile 6 y Mobile 6.5.

Es un software basado en la arquitectura cliente servidor, pero ya no solo porque necesitara de un servidor de Xgestevo con la base de datos como usa el software de

escritorio de Xgestevo, sino que necesita además de un servidor propio de movilidad sobre el que todos los terminales móviles se conectarán. Básicamente era un software que hacía consultas al servidor de movilidad y visualizaba resultados de las operaciones realizadas por este. Podemos ver un gráfico de la arquitectura en la sección 2.2.4

El servidor de movilidad fue el principal problema desde el principio, pues varios terminales conectados al mismo tiempo colapsaban el sistema y mezclaban información. El principal problema venia del hecho de que los terminales de Regumobile no realizaban realmente operaciones, sino que leían datos de los productos y los mandaban al servidor de movilidad que los procesaba y los devolvía al terminal para su presentación, por tanto, hacía falta un método de comunicación ágil, etiquetado y lo suficientemente robusto para que, si había alguna perdida de comunicación, no se perdieran datos.

Se probaron varios métodos como el paso de mensajes o los sockets, pero el método que definitivamente quedó instaurado fue el de generar una solicitud en XML e insertarla en una tabla de la base de datos llamada aaaa_pdadatxxx. Esta tabla tiene una entrada por cada PDA que hay en el sistema registrada que incluye su id, la última solicitud y su estado. De esta manera las PDAs ya no mezclaban información, ya que cada una insertaba en su fila, el servidor de movilidad recogía esta información y por orden de llegada las procesaba y devolvía la respuesta en esa misma línea. Por su parte la PDA que estaba a la espera de una respuesta solo tenía que consultar la línea que le correspondía a la espera de la respuesta del servidor.

Las funcionalidades del software son tres principalmente, aunque la de entradas llego un poco más tarde: consulta y modificación de artículos, servir pedidos de clientes y generar cola de entrada.

- La consulta de artículos, permite ver y modificar el stock y la ubicación de un artículo.
- Servir pedidos, permite servir todos los artículos de un pedido o varios pedidos. Para ello, dados los pedidos ya validados, ordena todo el material a buscar y te va mostrando en pantalla el siguiente artículo que debes ir a buscar, ordenado por su ubicación alfabéticamente.
- Generar cola de entrada lo que hace es ir generando una lista de todos los códigos introducidos y luego introducir por cada ítem de la lista una línea en la tabla fcepdxxx. Será el personal de compras el que revisará finalmente la cola y generará la entrada si todo es correcto.

A continuación, pueden verse algunas capturas de pantalla de la aplicación:

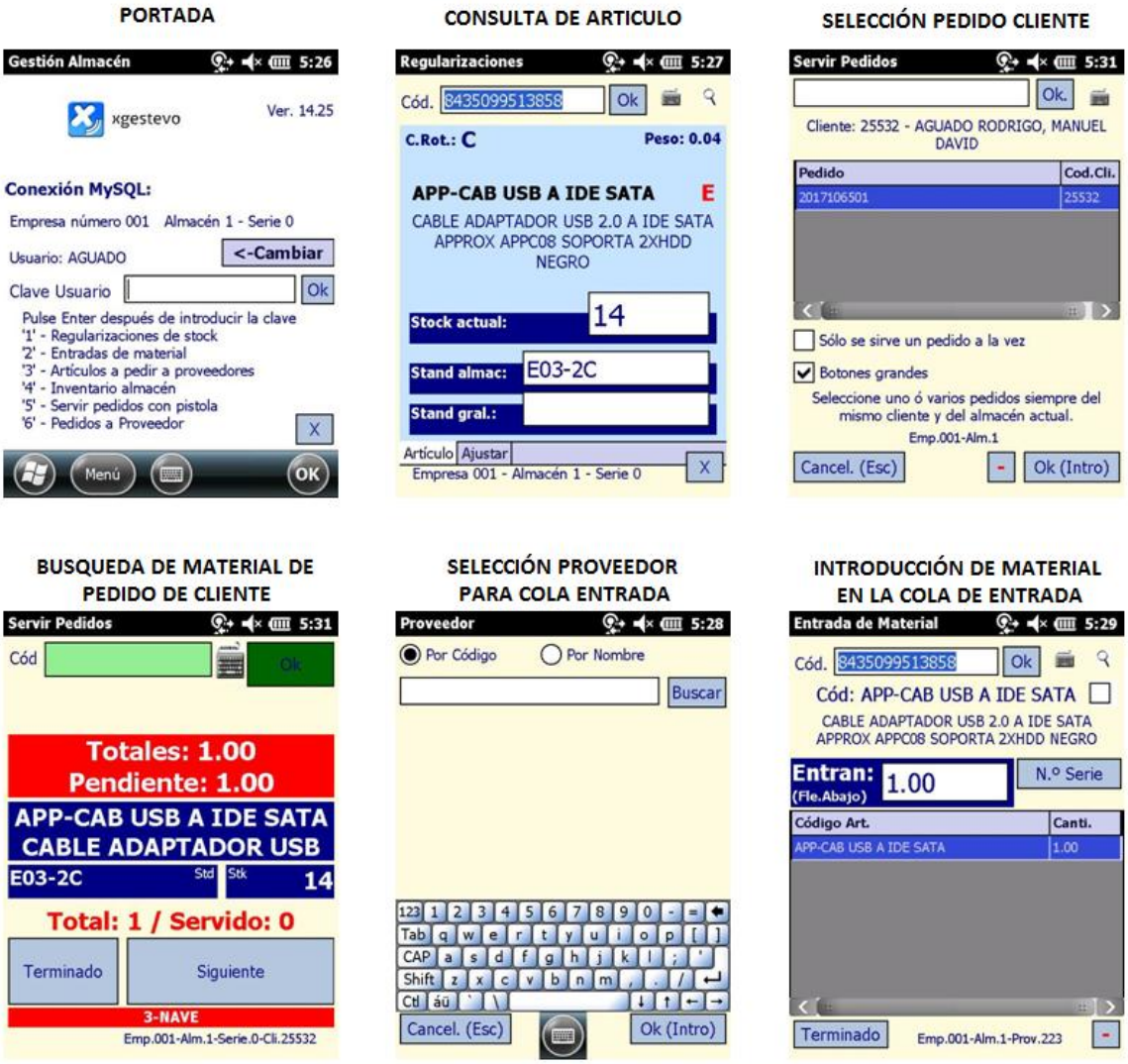

Ilustración 12. Pantallas de Regumobile

Es un software que aun en 2017 se sigue utilizando, en la versión 14.25 de la aplicación, pero es un software destinado a desaparecer en poco tiempo. Se ha alargado su tiempo de vida útil más de lo esperado, pues, aunque sigue funcionando relativamente bien, se encuentra con tres problemas muy graves:

El primero de ellos es que Microsoft sacó en 2010 Windows CE 6.0 en el Windows Phone 7, que es totalmente incompatible con CE 5, por lo que en la actualidad es un sistema operativo sin soporte y no tiene actualizaciones.

El segundo es la falta de disponibilidad de dispositivos. Al llevar un sistema obsoleto, los fabricantes cada vez hacen menos dispositivos con él, y por tanto los que quedan suben de precio y son cada vez más difíciles de conseguir. Por tanto, se añade la dificultad de reponer dispositivos rotos o ampliar la cantidad de puntos de trabajo.

El tercero es el tiempo y la evolución de las tecnologías. Xgestevo evoluciona, crece y cambia, pero Regumobile debido a que trabaja con una base obsoleta, cada vez se queda más atrás y es más difícil conseguir que se hagan actualizaciones complejas sobre el software. Por consiguiente, es solo cuestión de tiempo que llegue un punto en el que Regumobile no sea capaz de soportar un cambio de Xgestevo y deba quedarse atrás y desaparecer.

El software se encuentra disponible para su instalación en el propio programa de Xgestevo, siguiendo la ruta PDA/Web → Configuración de Dispositivos → Instalación de Software de PDA, seleccionando la opción PDA Regularizaciones (Windows Mobile). Al igual que los terminales de escritorio, el uso cada terminal móvil requiere de una licencia.

3.4.2 XgestMovil

Viendo que Regumobile se va quedando atrás, en 2015, los desarrolladores de Xgestevo empiezan a crear una aplicación en Android. El software se plantea para ser soportado por casi el 100% de los terminales, pues se establece como versión de base la 2.2.2 (Froyo).

Debido a determinadas decisiones de la dirección, se comienza a desarrollar primero un sistema de Preventa y autoventa que funcionará en modo offline. El dispositivo se sincroniza con la base de datos cuando se encuentra en las instalaciones de la empresa, cargando los datos de clientes y artículos en la memoria del terminal. El comercial visitará distintos clientes a lo largo de la jornada recogiendo pedidos y al final del día cuando vuelva a las instalaciones en las que se encuentra el servidor, volcará todos los datos en la base de datos y volverá a sincronizar.

A continuación, desarrollaron, esta vez en online, una funcionalidad que Regumobile no tenía, un sistema de gestión de Partes de trabajo, con el cual poder tener técnicos haciendo trabajos en la calle y que lleven actualizado al momento el estado de cada incidencia que deben solventar.

Actualmente se encuentra en la versión 22.23, y aunque se comenzó con el traspaso de Regumobile a XgestMovil, en la actualidad solo se puede consultar datos de clientes, proveedores y artículos. El resto aparece en desarrollo, pero actualmente se encuentra parado el proyecto y no se está avanzando en sus nuevas funcionalidades.

Aquí podemos ver algunas de sus capturas de pantalla:

Ilustración 13. Capturas de XgestMovil

XgestMovil sigue la misma arquitectura de conexión, en el modo online, que Regumobile. Se conecta a través del mismo servidor de movilidad, por lo que sigue insertando en la misma tabla aaaa_pdadatxxx instrucciones en XML y esperando a que el servidor procese los datos y se los devuelva. Sigue siendo un terminal de introducción de

datos y visualización, no se aprovecha todo el potencial de proceso que un terminal con Android podría conseguir, aligerando de esta manera la carga del servidor de la base de datos. Además, sigue usando el sistema de licenciamiento de terminales móviles que usa Regumobile.

El software podría haber sido una buena opción como sustituto para Regumobile, pero estando como se encuentra actualmente, no sirve para el control del almacén.

Puede encontrarse la aplicación en el siguiente enlace http://update.xgest.net/uploads/aplicaciones/XgestMovil.apk o usando el código QR

3.4.3 Xgestevo Movilidad

A comienzos de 2017, las empresas usuarias de Regumobile piden un cambio en el sistema, necesita modernizar su imagen y acelerar los procesos, además de la dificultad cada vez mayor de mantener los terminales antiguos. Se plantea la creación de un nuevo sistema de control de almacén para dispositivos móviles, se pretende aprovechar las tablets cada vez más ligeras y de pantalla más grande y los portátiles convertibles.

Para ello se comienza el desarrollo de Xgestevo Movilidad, un nuevo módulo de control de almacén, pero esta vez programado para navegadores web, de esta manera se puede ejecutar en cualquier terminal que pueda usar un navegador.

El planteamiento inicial del desarrollo es el de programar toda la aplicación en PHP con el framework Laravel 5, necesitando para su funcionamiento un servidor web Web2py server que se deberá instalar en un entorno de Windows desde el propio software de escritorio Xgestevo.

El software se encuentra aún en fases tempranas de desarrollo, aunque se ha podido acceder a alguna de sus funcionalidades en fase beta que ha proporcionado la empresa Xgestevo para testearlas e incluir algunas imágenes en este proyecto.

Ilustración 14. Inicio y login Xgestevo Movilidad

Tras algunas pruebas, se puede comprobar que la aplicación es totalmente responsiva y por tanto se adapta a cualquier tamaño de terminal en la versión y opciones que se han podido probar. También, Software de Gestión Xgest, S.L. ha proporcionado imágenes de todo el contenido que cada uno de sus tres apartados va a contener, pudiéndose ver que cubre la gran mayoría de las operaciones que día a día se hacen en un almacén.

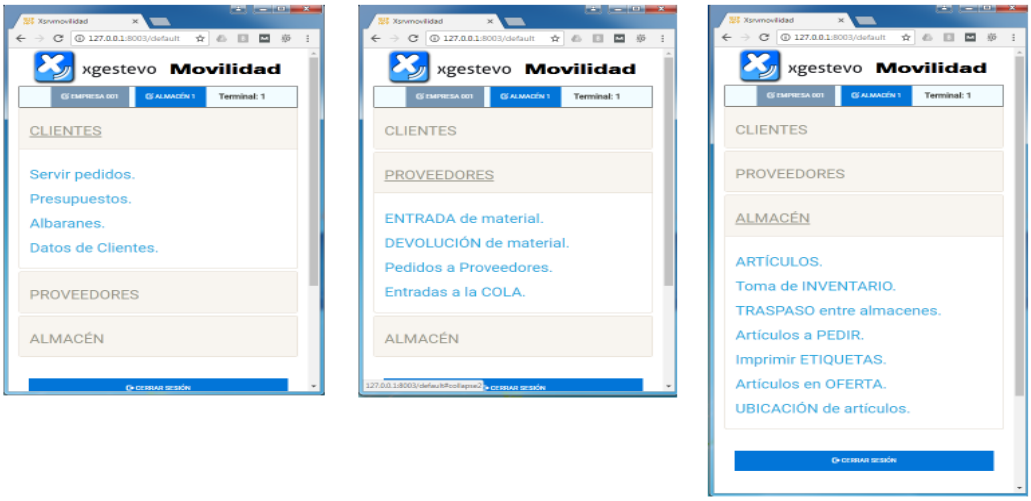

Ilustración 15. Menús principales de Xgestevo Movilidad

Como se ha dicho, hará uso de un servidor web, por lo que necesita de otra máquina para funcionar, como pasaba en las aplicaciones anteriores. También se ha confirmado que seguirá el mismo tipo de licenciamiento por terminal que haga uso de la aplicación.

3.4.4 Conclusiones

Como se puede ver, se disponen actualmente de tres tipos distintos de software de control de almacén que funcionen con Xgestevo. Aunque realmente, totalmente funcional solo es Regumobile. Tanto XgestMovil como Xgestevo Movilidad pueden ser, en un futuro cercano, buenos candidatos a sustituirlo, aunque aún se encuentren en desarrollo.

Los principales problemas que presentan los tres son los mismos: necesidad de un servidor que orqueste el funcionamiento de los terminales móviles, licencia por cada

dispositivo que queramos usar, cada modificación sobre el software que se necesite requiere de una solicitud a la empresa Software de Gestión XGEST, S.L. y por tanto requiere de un desembolso y de un tiempo de espera que puede verse dilatado en función de la carga de trabajo que tengan o del periodo del año en el que nos encontremos.

Además, como se ha visto en cada uno de los tres programas, cada uno tiene sus problemas específicos.

Regumobile es antiguo y cuesta mucho mantenerlo y obtener nuevas funcionalidades que se adapten a los nuevos métodos de trabajo de la empresa o por lo menos que lo hagan de manera eficiente. Los terminales son también cada vez más difíciles de encontrar, también los son sus piezas de repuesto y ambas opciones cada vez son más caras.

XgestMovil está creado en Android y orientado a terminales mucho más modernos, pero las funcionalidades actuales del software, no cubren con las necesidades del control de almacén, pues gran parte de ellas se encuentran por hacer y actualmente no se está desarrollando nada en esta plataforma. El proyecto se encuentra actualmente parado.

Xgestevo Movilidad es un proyecto en PHP, actualmente en fase de desarrollo y con betas con funcionalidades muy limitadas. Está orientado a terminales con posibilidad de instalación de periféricos como impresoras, pues usan los elementos locales del terminal que hace uso de ella para la impresión de etiquetas y documentos. Por tanto, está orientado sobre todo a pequeños portátiles o tabletas con Windows. Además, surge otro pequeño problema que aún no ha podido ser resuelto en todas las pruebas realizadas en terminales que no llevan teclado integrado, cada vez que se quiere escribir, aparece el teclado virtual y cubre gran parte de la pantalla. No debe olvidarse que el uso principal de estos programas va a ser con lectores de códigos de barras y por tanto se desea disponer de toda la pantalla para mostrar datos.

Este es el punto de partida en el que se plantea la creación de un software independiente, que de verdad sirva para cubrir las necesidades que los almacenes de las empresas que usan Xgestevo puedan usar de manera cómoda, optima, con terminales más baratos y no tan específicos como las actuales PDAs, con una nueva estética y con posibilidad de usar pantallas mucho más grandes. Pero, sobre todo, con la posibilidad de ampliación de nuevas características y separando el desarrollo de Xgestevo del módulo de control de almacenes.

No es posible el uso con Xgestevo de ningún otro software que no sea específico, pues la base de datos que almacena los datos tiene una estructura propia y requiere de profundos conocimientos de su funcionamiento ya que no dispone de relaciones, ni de procedimientos visibles, ni triggers que mantengan actualizados campos de otras tablas cuando se modifican tablas principales. Todo está hecho en el código de Xgestevo y no es accesible salvo para los desarrolladores de la empresa.

Como consecuencia de lo anteriormente dicho, no ha sido imposible acceder al código para conocer las rutinas, por lo que gran parte del trabajo realizado para el desarrollo de este software ha tenido que proceder del conocimiento del programa como usuario y administrador de la base de datos, además de realizar muchas pruebas para ver que campos se actualizaban en cada operación básica del sistema relacionado con el almacén.

4. Identificación de requisitos

A lo largo del capítulo anterior se han podido ver todas las opciones actuales para la gestión de un almacén con Xgestevo y cuáles son las funcionalidades mínimas y comunes que todas las alternativas tenían o en ocasiones de las que carecían, pero a continuación se enumerarán y especificarán cada una de ellas.

Además, tras varias entrevistas con clientes del programa, se han podido extraer algunas funcionalidades extras y que también son necesarias para este módulo.

4.1 Requisitos funcionales

Configuración.

- La aplicación debe interactuar directamente con la base de datos de Xgestevo, por lo que debe permitir configurar, probar y guardar la conexión directa con el servidor.
- La aplicación puede hacer uso de impresoras de etiquetas Zebra en red. Cada instancia de la aplicación se conecta a una impresora directamente y se tiene que poder configurar y probar desde la configuración de cada terminal.

Login.

- Se deben usar los mismos datos de usuario que se tienen en la base de datos de Xgestevo.
- Cada usuario tiene que hacer login con su usuario y contraseña de Xgestevo antes de poder usar cualquier función de la aplicación.
- El usuario tiene un almacén de inicio asignado por defecto y se carga con sus datos tras ser seleccionado, pero el usuario podrá cambiarlo en el momento de entrar en el sistema.
- El almacén seleccionado vincula todas las demás funciones de la aplicación a trabajar en dicho almacén.

Consulta de artículos.

- La consulta de artículos debe poder hacerse introduciendo cualquier código relacionado con el artículo: su código principal, cualquiera de sus diez códigos alternativos de la ficha de articulo o si fuera necesario, también de los códigos extras que se encuentran en la tabla de códigos especiales.
- Se deben mostrar todos los datos básicos para el trabajo en el almacén, que son: Su código principal, su descripción, el stock del almacén en el que se está trabajando, el

peso del producto, su categoría de rotación y el stand en el que está ubicado dentro del almacén.

- Para facilitar la identificación se debe añadir también una fotografía del producto.
- El peso, categoría de rotación y ubicación deben poder modificarse.
- La imagen del artículo debe poder cambiarse, tanto desde la memoria del terminal, como directamente haciendo una foto desde la cámara del dispositivo.
- Debe poder controlarse el stock del almacén y por tanto se tiene que poder modificar el stock del artículo que se está consultando y modificarse de manera efectiva como un albarán de regulación de stock.

Entradas de almacén.

- Se debe permitir la creación de colas de entrada del proveedor seleccionado.
- Debe permitirse la entrada directa de material en el almacén en el que se esté trabajando, con una entrada de proveedor que incremente el stock de los artículos.
- Como tercera opción se puede elegir servir pedidos pendientes que tenga el proveedor desde el material que se ha leído. El material sobrante que no estaba pedido, debe introducirse en una cola de entrada para su posterior revisión.
- La lectura del material se hará de uno en uno, pero debe poder modificarse esta cantidad manualmente si se desea introducir rápidamente varios artículos iguales.
- Si se produce un problema con el terminal, se debe poder continuar con el proceso de entrada tras la recuperación.

Servir pedidos de cliente.

- Se deben poder servir varios pedidos a la vez. Pero estos pedidos deben cumplir las mismas condiciones. Ser para el mismo cliente, del mismo almacén, ir al mismo centro o al mismo sub clientes, la forma de pago debe ser la misma y la agencia coincidir.
- El pedido puede servirse buscando uno a uno todos los artículos.
- Cuando se busca el material la aplicación debe ordenar los artículos por stand y generar una ruta que optimice la búsqueda.
- En el caso de la búsqueda, se mostrará solo el siguiente artículo que se debe ir a buscar, con su foto para la identificación y la cantidad que se pide y la que se ha servido.
- El pedido puede servirse después de tener todo el material ya preparado y sin necesidad de ir a buscarlo.
- Su un artículo no se encuentra, puede pasarse al siguiente.

- En caso de tener el material debe presentarse un listado con el material pedido y el servido, que se actualice según se introduzcan códigos.
- Si el artículo lo requiere debe poder suministrarse su número de serie.
- El proceso de servir material debe poder finalizar, aunque no esté todo el material.
- El proceso de servir material termina con la creación de uno o dos albaranes que disminuyen el stock del almacén.
- Si la empresa está obligada a la Inversión de Sujeto Pasivo, el cliente también y el producto es un ordenador, tablet, smartphone o consola, se debe crear un albarán en la serie 25. El resto del material se introduce en un albarán de la serie de trabajo 0.
- Tras el proceso de crear los albaranes, se pueden imprimir etiquetas por cada bulto.
- Las etiquetas de envío van ligadas a un albarán e incluirán los datos del cliente o sub-cliente, el número de bulto, peso aproximado y sobre el número de albarán en un código de barras en code128 en el formato necesario para que Xgestevo pueda remesar esos bultos.
- Si se produce un problema con el terminal, se debe poder continuar con el proceso de servir pedido tras la recuperación.

4.2 Requisitos no funcionales

Será necesario, además de los anteriores requisitos de funcionamiento del apartado anterior, el cumplimiento de unos ciertos requisitos adicionales de calidad al proyecto.

Documentación.

- Se debe proporcionar un manual de usuario, con la explicación de todas las funcionalidades del software.
- Listado de verificación rápida de errores. Un listado con todos los errores conocidos que pueden producirse y su solución aconsejada.
- El código debe ir lo suficientemente bien comentado para que sea fácil continuar y ampliar el proyecto.

Compatibilidad.

- El software debe ser compatible tanto para tablet como para smartphone.
- La versión del Android mínima debe ser la menor posible, para dar la mayor retro compatibilidad posible.
- Debe poder funcionar en cualquier entorno de red.

Seguridad.

- La aplicación solo debe usarse por usuarios autorizados en el sistema Xgestevo.

Usabilidad.

- La aplicación debe ser concisa y solo mostrar los datos necesarios para realizar el trabajo para el que está diseñada.
- La apariencia debe seguir los patrones de estilo de Xgestevo
- Está destinada a trabajos en almacén, con la menor interactuación con la pantalla.
- Cualquier actuación sobre la pantalla debe ser clara y con áreas grandes de actuación. Botones grandes y evitar iconos pequeños y difíciles de pulsar.

Rendimiento.

- La aplicación debe ser lo más ágil posible, pues el rendimiento del personal del almacén dependerá de lo rápido que sea el software.
- Trabajar con las menores conexiones posibles a base de datos central. Para evitar sobrecargar el sistema y evitar retrasos en el proceso de datos.

Estabilidad.

- La aplicación debe ser capaz de gestionar todos los fallos que puedan surgir en el proceso normal de funcionamiento.
- Ante cualquier fallo no deben introducirse errores en la base de datos Xgestevo. En caso de ser inevitable, debe avisarse para su corrección manual.

4.3 Metodología de desarrollo

Debido a la clara especificación de los requisitos del sistema, a la clara separación en módulos de funcionalidad independiente, es fácil la utilización de un modelo incremental de desarrollo, en el que en cada iteración del ciclo de trabajo, se implementa un modelo en cascada completo que desarrolla cada módulo independiente, y en él, se analizan (A) las necesidades del nuevo módulo, se diseña (D), se codifica (C), se crea un prototipo que se prueba (P) y se integra (I) en la solución final. Finalmente, cada fin de iteración hace que el prototipo final vaya formándose y provoca que cada módulo finalizado entre en una fase de mantenimiento (M) que irá depurando el sistema hasta la finalización del proyecto.

En el siguiente gráfico se muestran las distintas iteraciones que se han realizado:

Tabla 2. Iteraciones del modelo de desarrollo incremental del proyecto

Configuración. Conexión base de datos	A	D	C	P	I					M					M					M					M				M
Login de usuarios						A	D	C	P	I					M					M					M				M
Consulta y modificación de artículos											A	D	C	P	I					M					M				M
Entradas de proveedor																A	D	C	P	I					M				M
Servir pedidos de clientes																					A	D	C	P	I				M
Configuración. Imprimir etiquetas de envío																										A	D	C	P	I

Cada una de estas iteraciones se desarrollará más adelante cuando se vea la descripción del proyecto y la posterior codificación de cada uno de los módulos.

4.4 Herramientas utilizadas

Para el completo desarrollo de este proyecto, ha sido necesaria la utilización de varias herramientas que a continuación se describen.

Android Studio 2.3.1. Es el Entorno de Desarrollo Integrado (IDE) oficial para la creación de proyectos en Android. Está basado en la IntelliJ IDEA (JetBrains s.r.o., 2017). Ha sido el entorno en el cual se ha codificado todo el código de este proyecto. Puede encontrarse en Google. Android Studio. (2017)

MariaDB v5.5.29. Es un fork libre y gratuito de MySQL que es la base de datos por defecto de Xgestevo desde hace algunos años. Esta base de datos almacena todos los registros que se deberán consultar, modificar e insertar. Puede encontrarse en MariaDB. (2016)

HeidiSql v9.4. Es un software Opensource con licencia GPL, cuya principal función es la de servir de herramienta de diseño y manipulación de bases de datos basadas en SQL. En esta herramienta se crearán las instrucciones de manipulación de MariaDB necesarias para este proyecto. Puede encontrarse en Ansgar Becker. (2017).

Xgestevo v31.46. Software de gestión para el que se ha desarrollado este programa. Puede encontrarse en Saura, D. (2015)

Git v2.13.2. Es un software de gestión de versiones que se ha utilizado para el desarrollo del proyecto. Totalmente compatible con el entorno de desarrollo Android Studio y con el sitio de publicación del código SourceForge. Puede encontrarse en Git. (2017).

SourceForge. Es un repositorio online donde se publican cientos de programas de código abierto. En este repositorio se encuentra publicado el código de este proyecto. Puede encontrarse en SourceForge. (1999).

ZebraDesigner v2.5. Es una herramienta gratuita proporcionada por Zebra que se utiliza para diseñar etiquetas y posteriormente exportarlos. Para este proyecto se ha usado para el diseño de las etiquetas de envío y posteriormente se ha generado el código ZPL asociado a dicha etiqueta para que puedan entenderlo las impresoras Zebra. Puede encontrarse en Zebra. (2017).

5. Descripción del proyecto

El objetivo general de este programa es el de ser un módulo con el cual se realicen las tres principales actividades dentro de todo almacén, que son: la entrada de material, el control de stock y la salida de material. Para ello, el personal del almacén deberá trabajar con unos terminales móviles en Android, previamente configurados por técnicos del sistema para permitir la interactuación de estos terminales con la base de datos de Xgestevo.

Para la realización de esta actividad, el software se dividirá en cinco bloques principales, que serán los que se desarrollarán para el correcto funcionamiento del sistema:

- Configuración del sistema
- Login de usuario
- Consultar artículos
- Entrada de material
- Servir pedidos

Cada uno de estos bloques será desarrollado en los siguientes sub secciones sin entrar en detalles del código, pero si con una explicación detallada de que deberá hacer cada bloque y la manera en la que se debe interactuar con el sistema de Xgestevo.

5.1 Configuración del sistema

Esta sección será la primera a la que se deberá acceder tras su instalación antes de poder trabajar con este programa, ya que, para cualquiera de las demás funcionalidades, es necesaria la interactuación con una base de datos de Xgestevo en funcionamiento. La única manera de realizar esta conexión con la base de datos es conociendo previamente los patrones de configuración, que tendrán que ser conocidos por el personal técnico que realice la instalación.

Los elementos necesarios para la configuración de la conexión son los siguientes:

- Dirección de la base de datos
- Nombre de la base de datos
- Usuario
- Contraseña
- Puerto de conexión
- Tiempo máximo de espera

Los primeros cinco elementos son los únicos necesario y los que se deben pasar al JDBC de MySQL para poder hacer cualquier conexión con esta. Dicho JDBC facilitará en

gran medida toda interacción con la base de datos, pues proporcionará todo el repertorio de instrucciones del que se dispone en la versión de la base de datos con la que se trabaje y no será necesario programar una a una estas interacciones básicas, como pueden ser los select, update y delete.

Esta librería se podrá obtener en todo momento de manera gratuita de la web oficial del motor de la base de datos con la que trabajemos (MySQL, 2017).

El último parámetro, el de tiempo máximo de espera de la base de datos, será un elemento opcional, aunque recomendado. Su función será la de establecer un tiempo máximo que la aplicación estará esperando una respuesta por parte de la base de datos a cualquier interacción que se realice con ésta. De esta manera se logra evitar esperas inacabables que se pueden provocar cuando se bloquea la base de datos durante una operación, o se pierde la respuesta de esta por culpa de perder la señal en el terminal o cualquier otro impedimento que pueda provocar que no finalice correctamente la comunicación entre ambos extremos.

5.2 Login de usuario

Este proceso será por el primero por el que cada usuario deberá pasar cada vez que comience a usar el terminal móvil y su función es la de identificar con usuario y contraseña a cada persona que quiera usar la aplicación. Este usuario se usará para marcar cada operación realizada de modificación en la base de datos.

Primeramente, se elegirá al usuario de una lista de usuarios que hay en el sistema en la tabla fcusu. El sistema comprobará la empresa y almacén por defecto en el que trabaja ese usuario y lo asignará, aunque permitirá modificar cualquiera de ellos si ese usuario tiene permiso para cambiarlos. Finalmente, se escribe la contraseña y comienza el proceso de identificación

El proceso de identificación consiste en enfrentar la contraseña que se supone que pertenece a su usuario, con la que la base de datos guarda encriptada. Para poder realizar esta comprobación, se encripta la contraseña que escribe el usuario antes de hacer la comparación.

El proceso de encriptado de contraseñas que usa Xgestevo consiste en coger cada carácter de la contraseña, convertirlo a su equivalente en ASCII (Microsoft, 2005) decimal y restarle la posición ordinal que ocupa dicho carácter en la contraseña.

Por ejemplo, para una contraseña con la palabra clodd18, su encriptación sería:

Tabla 3. Proceso de encriptación de contraseña de Xgestevo

Contraseña	c	l	o	d	d	1	8
Equivalente ASCII	99	108	111	100	100	49	56
Ordinal	1	2	3	4	5	6	7
Equivalente ASCII de la suma	98	106	108	96	95	43	49
Encriptado	b	j	l	`	_	+	1

Si el proceso de identificación es correcto, la aplicación permitirá el acceso a las demás opciones del sistema. En caso de no ser correcto, dará un error y no permitirá el acceso a las demás funcionalidades.

5.3 Consultar artículo

Esta función consta de dos partes, por un lado, la consulta de artículos, y, por otro lado, una vez cargados estos datos, poder modificarlos si alguno está mal.

5.3.1 Consulta

En la primera parte, que es la de consultar artículos, el usuario leerá con un lector de códigos de barras el código que identifica a un artículo.

Dado que cada artículo en la base de datos puede estar identificado por varios tipos distintos de códigos, se tendrá que comparar con todos los campos de cada artículo en la tabla fcartxxx que son un total de 11 (código del articulo más 10 códigos alternativos) y además se deberá verificar con la tabla de códigos alternativos extras fccltxxx, que no hay ninguna entrada que corresponda con este código.

Si existe una coincidencia, será única, pues el sistema Xgestevo, no permite introducir códigos repetidos en ningún caso.

Si no encuentra coincidencia, mostrara un aviso de: “Articulo no encontrado”. Pero en el caso de hallar una, traerá el artículo a la pantalla y nos mostrará los siguientes atributos:

- una imagen del articulo si existe, traída de la tabla fcfosxxx
- el código del articulo
- descripción del articulo
- peso
- la categoría de rotación

- su ubicación en el almacén
- el stock traído de la tabla fcstkxxx

Casi todos los valores están en la propia tabla fcartxxx, a excepción de dos. La foto es el primero de los valores que no está en fcartxxx, pero está en fcfosxxx. Y que para su modificación se podrá usar una de la galería o tomar una foto con la cámara del terminal.

El otro valor que no se guarda en fcartxxx es el stock del almacén en el que se encuentra trabajando el terminal. Existe un campo stock en fcartxxx, pero es el stock total de todos los almacenes, y no sirve para el control de inventario del almacén en el que se está trabajando que se pretende seguir con esta parte de la aplicación. Para ello se debe de traer el registro del stock del artículo que se está consultando y del almacén en el que se trabaja.

5.3.2 Modificación

En caso de detectar que alguno de esos datos sea incorrecto, existe la posibilidad de entrar en la pestaña de modificación, en la que cambiar cualquiera de estos valores. Como se ha visto, casi todos están en la tabla fcartxxx y simplemente hay que cambiar su valor y guardar los cambios. Para la foto, el proceso es algo distinto, para su modificación se podrá usar una de la galería o tomar una foto en el mismo instante con la cámara del terminal.

Para el cambio del stock el proceso es algo más complejo, ya que no vale simplemente con cambiarlo, pues Xgestevo se basa en un sistema férreo de control de stock en el cual, el material siempre entra al almacén a través de una entrada de proveedor y sale a través de un albarán de cliente. Por tanto, no sirve simplemente cambiar el campo en fcstkxxx, pues, cuando cualquiera de las rutinas de recálcalo de stock revise el número de entradas y de albaranes, volverá a corregir cualquier cambio que se haga en esta tabla.

La forma correcta de modificar el stock en Xgestevo es usando un albarán en una ficha de cliente que se establece con la creación de cualquier empresa y que se encuentra en la segunda tabla de configuración general fcge2xxx. El proceso de regulación es sencillo, se introduce el valor del stock real, se calcula la diferencia entre el stock que había y el que se ha introducido y el resultado de esa diferencia será la cantidad que se deberá añadir en la línea de albarán que se introducirá en fcliaxxx. Además, como Xgestevo no dispone de ningún sistema de triggers, se tendrá que actualizar también fcstkxxx con el nuevo stock y el campo de stock de fcartxxx, añadiendo al stock que había, el resultado de la resta anterior.

Xgestevo solo realizará un albarán de regularización de stock al día y por usuario, por lo que, si ya existe una cabecera en ese día para ese usuario, solo añade una línea más. En caso de no existir la cabecera, la creará antes de crear la línea de regulación.

5.4 Entrada de material

En esta funcionalidad se dispone de tres opciones para realizar una entrada. En la primera se mantiene la funcionalidad original de entradas de Regumobile en la cual se pasa el material que se desea que entre y el sistema lo pasa a una cola que posteriormente el departamento de compras revisará y convertirá en una entrada de material que añadirá stock al almacén.

La segunda opción a la hora de hacer las entradas será la de generar una entrada directamente desde el listado de material que se ha introducido en el terminal móvil.

Por último, la tercera opción será la de procesar todo el material que se ha introducido en el terminal móvil y generar una entrada solo del material del que se disponga de pedido de proveedor y pasar a cola el resto de material.

5.4.1 Lectura de material para entrada

En cualquiera de las tres opciones el proceso inicial será el mismo, primero se elige el proveedor sobre el que se trabaja, para ellos se introduce o el código de proveedor o buscando una cadena de texto que coincida con parte del nombre del proveedor, esta consulta se realizará sobre la tabla fcproxxx y mostrará el listado de proveedores que cumplen con los patrones de búsqueda y que no se encuentran bloqueados.

A continuación, viene el proceso de introducción de material de la entrada, en el cual se aplicará el mismo concepto que en la consulta de material. Se va introduciendo uno a uno los códigos del material que se quiere procesar y, tras la verificación de que existe ese material dado de alta en la base de datos, se introduce una línea en el listado.

Una vez acabada la introducción del material, se procesará de la manera que se haya elegido:

5.4.2a Generar cola de entrada de proveedor

Se recorre todo el listado generado por la introducción de los códigos y por cada artículo se introduce una línea en la tabla de la cola de la PDA fcepdxxx en la que se indica entre otras cosas el articulo y la cantidad total que entra de éste.

Estas colas serán procesadas a posteriori por el personal de compras en el software de escritorio de Xgestevo y generará manualmente una entrada con los elementos de la cola que necesiten.

5.4.2b Generar entrada de proveedor

En este caso se debe crear una cabecera de entrada previa al proceso de introducir el material. Esta cabecera llevará los datos del proveedor, su forma de pago y el coste total de la entrada entre otros datos en la tabla fccbexxx.

A continuación, se recorre el listado generado por la introducción de los artículos y por cada línea se crea una línea en la tabla fcliexxx, con todos los datos del artículo, su cantidad y su coste medio.

Por cada línea que se vaya añadiendo hay que actualizar el stock general de la tabla fcartxxx y la tabla de stock específico del almacén en el que el terminal está trabajando en fcstkxxx.

5.4.2c Generar entrada de proveedor desde pedido de proveedor

Este es un proceso que mezcla los dos anteriores, pero que además introduce el uso de los pedidos del proveedor.

Previo a la generación de la entrada, para este caso en particular se va asignado el material, según se va introduciendo, a cada pedido pendiente, por orden de antigüedad, de tal manera que se vaya descontando el material de este listado según las líneas de pedido de fclopxxx se vayan completando o ya no quede cantidad suficiente para llenar una línea y esta se complete parcialmente.

Ahora se genera la entrada, para lo cual, en este caso, se cogen las líneas de pedido para las cuales se ha asignado material. En caso de tener alguna línea tras llegar a este punto, se generará una cabecera de entrada en el proveedor en la tabla fccbexxx y tantas líneas en fcliexxx como de líneas agrupadas se tenga, con la cantidad total de artículos que entran en cada una y a su precio de pedido. El último paso de la creación de la entrada es la de sumar el importe de todas las líneas de entrada y actualizar los valores de coste de la entrada.

Tras la creación de la entrada, se actualiza el estado de los pedidos de proveedor. Para ello se liquidan las líneas de fclopxxx para las que se haya servido todo el material, poniendo cantidad servida igual a cantidad pedida y marcando el campo de liquidado a si o, en su defecto, si la línea no ha podido servirse completamente, se actualizará el campo de cantidad servida igual a la cantidad que haya podido asignarse y el campo de liquidada permanecerá a no. Para toda línea totalmente liquidada, se debe chequear si quedan más líneas pendientes en ese pedido, de no ser así, la cabecera del pedido de proveedor en fccopxxx deberá marcarse también como liquidado a sí.

Cabe la posibilidad de que, en el listado original del material que entra, quede aun material sin asignar tras la creación de la entrada. Es el caso del material que ha llegado, pero no había un pedido de proveedor. En este caso se actuará como en el primer caso y se creará una cola de material pendiente de entrada. Este material será revisado por el departamento de compras y, si lo estima oportuno, creará una entrada manual para hacer que ese material aparezca en el stock.

Ya solo queda actualizar el stock del material que entra en la tabla fcstkxxx perteneciente al almacén en el que se esté trabajando, para lo cual simplemente se suma el material para el que se ha hecho entrada al valor que actualmente se almacena en la tabla de stock. También se tendrá que actualizar el stock general que hay en la tabla fcartxxx.

5.5 Servir pedidos

Este es, de los cuatro procesos, el más complejo por la gran cantidad de variable a tener en cuenta, porque hay dos formas de servir los pedidos y porque finalmente también se deberá generar una etiqueta de envío que se pasará a una impresora.

Al igual que pasaba en las entradas lo primero que se debe hacer será elegir la forma en la que servir el pedido o los pedidos. Se contemplan dos, la primera servir buscando material, la segunda será servir material localizado. Ambos casos se verán más detalladamente más adelante.

5.5.1 Cargar pedidos

Una vez decidida la forma de servir material se pasa a un método común que consiste en introducir los números de pedido que se desea servir. Puede ser que se sirva un solo pedido, en cuyo caso se pasará a servir el material, o puede que se quiera servir más de un pedido. En este último se introducirán uno a uno todos los pedidos que se quieran servir, pero deberán cumplirse ciertas condiciones para poder servirse juntos y generar un único albarán:

- Los pedidos deben ser todos del mismo almacén.
- El cliente debe ser el mismo en ambos pedidos.
- Deben tener la misma agencia de envío.
- Si el cliente tiene varios centros de envío, todos los pedidos deben pertenecer al mismo, pues la dirección de destino debe ser la misma.
- La forma de pago de los pedidos debe ser la misma.
- Si el pedido es para un sub-cliente, es decir, se envía a una persona distinta de la persona a la que se factura, tampoco podrá servirse con otros pedidos. Esto se comprueba en la tabla fcdrexxx, asegurándose que este pedido no tenga ningún registro en dicha tabla.

Cualquiera de estas situaciones dará un error al leer el código del pedido y no dejará introducirlo en el sistema.

A excepción de la tabla de sub-clientes fcdrexxx, el resto de información para verificar los pedidos se saca de la cabecera de estos en la tabla fccocxxx. Esta comparación se hace entre el primer pedido y cada nuevo pedido que se introduce.

Una vez se tienen todos los pedidos y se selecciona el empezar a servir, se dan dos situaciones distintas en función de la selección que se haya hecho para servir el pedido.

5.5.2a Servir pedido buscando material

En este caso no se sabe en donde se encuentra el material, por lo que el terminal móvil deberá guiar al usuario por la ruta más rápida para ir recogiendo todo el material que hace falta para terminar el pedido.

Antes de entrar a esta ventana para servir material, el terminal ha tenido que cargar todas las líneas de fclocxxx pertenecientes al pedido o pedidos, ha consultado por cada uno en la tabla de artículos fcartxxx su ubicación y ha ordenado por este campo alfanumérico por el que está organizado el almacén.

Tras la ordenación mostrará una ficha del siguiente artículo a buscar. En esta ficha se incluirá:

- Código del artículo.
- Primeras palabras de la descripción, hasta completar una línea.
- Foto para su correcta identificación.
- Ubicación
- Cantidad pendiente de servir
- Cantidad servida

A partir de este momento se comienza a servir el material leyendo su código de barras y comparándolo con cualquiera de los posibles códigos alternativos, si no coincide con ninguno, dará un error. En caso de que la tabla de artículos diga que éste necesita número de serie, aparecerá otra zona en la que deberá introducirse. Si la introducción del código es correcta, descontará uno de la cantidad pendiente y sumará uno a la cantidad servida.

Cuando se hayan servido la cantidad necesaria de un artículo, se saltará automáticamente al siguiente de la lista. En caso de no encontrar el producto se puede pulsar el botón siguiente y se salta al siguiente artículo de la lista que aún no esté servido totalmente.

Una vez se han servido todos los productos y ya no muestra ninguno para servir, o no se encuentran los que quedan pendientes, se pulsará el botón terminar que pasa al proceso de creación del albarán.

5.5.2b Servir pedido con material localizado

En este caso el material ya se tiene localizado y listo para pasarse por el terminal móvil, interesa en este caso que se haga de la manera más rápida posible y sin necesidad de seguir ningún orden.

El terminal móvil también carga en este caso todas las líneas de fclocxxx que tenga el pedido o los pedidos a servir. Pero en esta ventana muestra un listado con todos los códigos de artículo, la cantidad pendiente y la cantidad servida.

El usuario no tendrá más que ir cogiendo uno a uno los artículos y leerlos con el escáner y se irán descontando de la cantidad pendiente. En caso de leer un código erróneo o leer más cantidad de la que hay en el pedido, se mostrará un error.

Tras la introducción de todo el material del que se disponga, se procederá a pulsar el botón de terminar y empezara el proceso de crear el albarán.

5.5.3 Creación del albarán

Tras la introducción del material se consulta si la empresa trabaja con Inversión de Sujeto Pasivo consultando en la tercera tabla de configuración del sistema fcge3xxx, de ser así, se consulta en fcclixxx si el cliente esta acogido ha dicho tratamiento del IVA. Si no se cumple alguna de las condiciones solo se creará un albarán, pero si se cumplen ambas, se debe separa el material en dos grupos: el del material sujeto a la Inversión y el que no está sujeto a la Inversión. Si un material está sujeto o no a la inversión se comprueba o en su ficha en fcartxxx o en su familia de ventas fcfcpxxx.

Si al final hay realmente dos grupos, se crearán dos cabeceras de albarán en fccbaxxx, uno en la serie del pedido para el material no sujeto a la Inversión y otro en la serie de documentos por defecto para la Inversión de Sujeto Pasivo, que será la 25. Esta serie puede localizarse en la tabla fcserxxx. En el caso de que los pedidos sean para un sub-cliente, también se deberá crear una copia, por cada albarán, de la línea en la tabla de direcciones de envío fcdrexxx asociada al envío y vinculándola con el número de albarán creado.

A continuación, se crea una línea de albarán en fcliaxxx por cada código de artículo y por cada precio de ese artículo, se le asigna la cantidad de artículos que han introducido de

cada uno y su precio. En el caso de existir varios precios para un mismo artículo, se empezarán a servir los pedidos más antiguos, por lo que antes se introducirán éstos.

Tras la creación de todas las líneas se procederá a actualizar la cabecera de cada albarán con la suma de los importes y costes de las líneas y se procederá al cálculo del IVA correspondiente.

Finalizada la creación de los albaranes, se actualiza el estado de las líneas de los pedidos de los que se han procesado. Para ello aumentamos la cantidad servida con la cantidad que se ha introducido, hasta llegar a la cantidad pedida como máximo o hasta que no quede más material que asignar. En caso de existir más de una línea por artículos, se procesa la más antigua. Si la cantidad servida llega a ser igual que la cantidad pedida, se marcará esa línea como liquidada.

Tras recorrer todas las líneas y actualizarlas, se revisa si todas las líneas de un pedido están liquidadas, y de ser así, se liquida la cabecera del pedido.

Por último, el stock de cada artículo es actualizado en la tabla de stock correspondiente al almacén en el que se está trabajando y el stock general de la ficha del artículo descontando en ambos casos la cantidad servida a la que hay en ese momento.

5.5.4 Generar la etiqueta

Aparece en este momento una nueva vista que dice que ya se han generado los albaranes y muestra su número y además pide un número de bultos que se van a enviar al cliente.

Al introducir el número y pedirle que lo imprima se actualizará la cabecera del albarán o albaranes con ese número de bultos y además generará una etiqueta con formato ya establecido y en condigo ZPL (Zebra, 2016), que es el lenguaje nativo que entiende la impresora Zebra para poder imprimir las etiquetas.

Estas etiquetas sacaran todos los datos de tres sitios distintos:

- De la cabecera del albarán fccbaxxx para el código de cliente, el número de etiquetas a imprimir y el código de la agencia de envío.
- De la ficha del cliente fcclixxx todos los datos de entrega del bulto, a no ser que se envíe un centro distinto al 0, en cuyo caso se deberá sacar de la tabla de centros fccenxxx asociados a ese cliente; o el tercero de los casos, si el pedido tiene asociado que se envía a un sub-cliente, se deberá de enviar a la dirección a los datos asociados al albarán en la tabla de direcciones de envío fcdrexxx

De la tabla de agencias de transporte fcvehxxx se extrae el nombre de la agencia de transporte.

6. Descripción técnica

Como ya se ha dicho anteriormente, este proyecto es un software para dispositivos Android, y como todo software para Android, lo primero que debe hacerse a la hora de crear un nuevo proyecto es elegir la versión mínima de SDK que podrá ejecutarlo. En este caso se ha decidido por poner como versión mínima API 14. Android 4.0 Ice Cream Sandwich. Existen varias razones para esta decisión, la primera, porque es la versión mínima que el SDK de Zebra recomienda para la utilización de su librería; además, debido al requisito de usabilidad, donde se pide que la aplicación sea lo más simple y clara posible, no se harán uso de grandes prestaciones que hagan necesario el aumento de versión mínima; pero, sobre todo, se cumplirá el requisito no funcional de compatibilidad, en el cual se pide la mayor compatibilidad posible con los dispositivos que hay en el mercado. Puede verse en Google. Android developer, versiones. (2017) que, en junio de 2017, el 99.2% de los terminales que hay en el mercado mundial, ejecutan como mínimo Android 4.0

A la hora de crear el proyecto se decide continuar con la nomenclatura actual de los nuevos módulos de Xgestevo, en la cual se añade una X al uso que se le va a dar al módulo, por lo que la aplicación de este proyecto se llamará **XAlmacen.**

Se definen también al comienzo de la aplicación unos patrones generales de estilo, que básicamente consisten en la aplicación a todas las vistas de un fondo (atributo background de la vista de cada actividad) con un degradado entre dos tonos de azul, para seguir también con la estética propia de Xgestevo:

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android">
    <gradient
        android:endColor="#3333ff"
        android:startColor="#81bef7"
        android:angle="270" />
</shape>
```

También se incorpora, al inicio del proyecto, el icono que se mostrará cuando se instale la aplicación. Dicho icono es el mismo que actualmente se utiliza en Xgestevo en su versión de escritorio y ha sido proporcionado por el equipo de desarrollo de Xgestevo y redimensionado a las distintas resoluciones que pide Android Studio 48x48, 72x72, 96x96, 144x144, 192x192.

Ilustración 16. Icono Xgestevo y XAlmacen

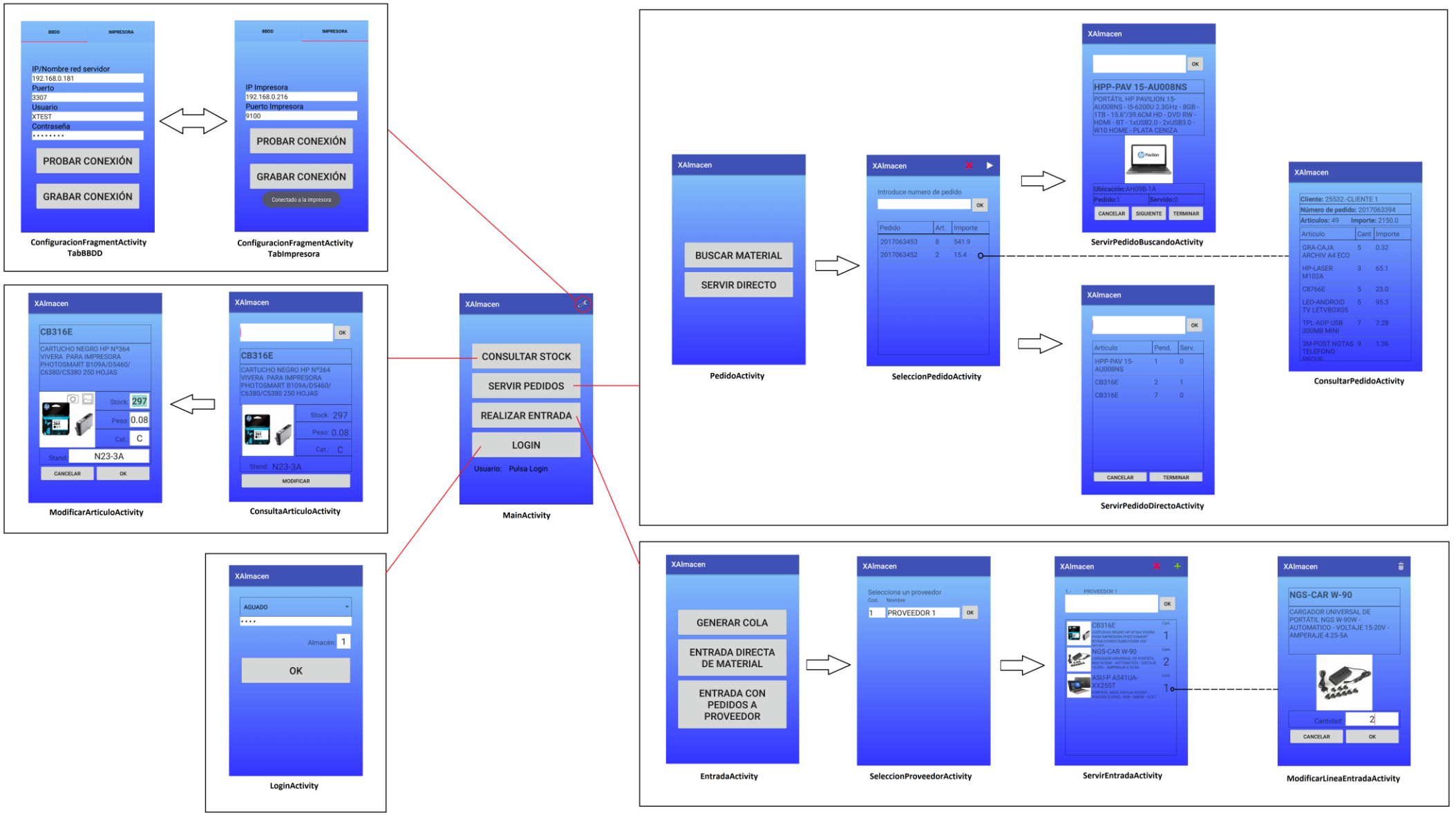

Ilustración 17. Actividades de la XAlmacen

6.1 Menú principal de la aplicación

6.1.1 MainActivity

La aplicación comienza en su clase y actividad principal, la MainActivity, en ella se mostrarán 4 botones, cada uno de los cuales lleva a una de las 4 principales funcionalidades del sistema: Consultar artículo, Servir pedidos, Realizar entrada y Login. Cada uno de estos 4 botones lanza a su vez una actividad diferente que será la encargada de llevar el hilo conductor de cada una de estas funcionalidades y que se verán un poco más adelante.

Otro elemento de esta actividad es el **usuario** que está usando la aplicación. Se define como variable *public static* para poder ser accedida desde toda la aplicación, junto con el **almacén** de trabajo.

Se hace, por último, uso de un recurso *menu* para añadir en la esquina superior un botón que lanzará la quinta de las actividades del proyecto, la configuración de la aplicación.

6.2 Configuración

Lo primero que se debe hacer nada más instalar la aplicación es entrar en las opciones de configuración para poder conectar a la base de datos e impresora de etiquetas.

6.2.1 ConfiguracionFragmentActivity

Esta actividad es la que se inicia tras pulsar el botón de configuración del menú de MainActivity. Esta actividad es de tipo *FragmentActivity* y el motivo para esta elección es que se ha decidido mostrar dos tipos de configuraciones, la de la base de datos por un lado y la de la impresora por otro. Para poder mostrar ambas en una vista se recurre al uso de Tabs, para cuyo uso necesitamos una clase de tipo *Fragment* por cada pestaña que se desee mostrar.

Por tanto, nada más iniciar ConfiguracionFragmentActivity, lo primero que se produce es la carga del layout que muestra la actividad, activity_configuracion_fragment y que solo contiene un *TabWidget* donde mostrar la Tabs seleccionable y un *FrameLayout* en donde se muestre el contenido de la Tab activa. Tras la carga del layout, se crea un *FragmentTabHost*, en el cual se añaden dos Tabs con la carga de cada una de las clases que deben ser vistas en pantalla.

```
tabHost = (FragmentTabHost) findViewById(android.R.id.tabhost);
tabHost.setup(this, getSupportFragmentManager(), android.R.id.tabcontent);
tabHost.addTab(tabHost.newTabSpec("TabBBDD").setIndicator("BBDD"),
        TabBBDD.class, null);
tabHost.addTab(tabHost.newTabSpec("TabImpresora").setIndicator("Impresora")
          ,TabImpresora.class, null);
```

El resto de funciones que aparecen en esta clase son las necesarias para operar con cada una de las Tabs y se verán en los siguientes apartados.

6.2.2 TabBBDD

Esta clase es uno de los Fragment que carga una de las pestañas que componen la actividad ConfiguracionFragmentActivity. Es la que se encarga de la configuración de la conexión con la base de datos de Xgestevo.

La función principal de esta clase es la de cargar la vista de la Tab cuando se hace clic en ella, cargando el layout asociado, fragment_tab_bbdd.xml. Contiene una única función, además de las necesarias para la creación de la clase y la vista. Esta función es la de cargarConfiguracionBBDD() y se encarga de acceder al archivo de preferencias de la aplicación y obtener los parámetros almacenados que permiten conectar a la base de datos.

Con la vista ya creada, se devuelve el control de la aplicación a ConfiguracionFragmentActivity, en donde se encuentran los demás métodos que actúan sobre la vista.

Existen dos métodos que actúan contra la TabBBDD, y son los de guardarConfiguracionBBDD(), que se encarga de, una vez pulsado el botón de guardar, coger todos los elementos que hay en los *EditText,* de la vista actual y guardarlos en el archivo de configuración de la aplicación; y el segundo método es el de conectar(), que se encargará de, una vez pulsado el botón de conectar, ejecutar una AsyncTask que se encargará de conectar con la base de datos Xgestevo.

Se decide usar una AsyncTask porque Android no permite hacer operaciones de red que puedan dejar bloqueado el hilo principal de la aplicación, por lo que se decide generar otro hilo secundario. Este hilo secundario se ejecuta con un tiempo de espera máximo de 30 segundos, esto se hace para evitar que, si la base de datos no responde, el hilo secundario permanezca ejecutándose indefinidamente. En caso de exceder el límite de tiempo se captura una excepción en la que se advierte de esta situación. Además, se permite que el AsyncTask sea cancelable, lo cual permite que el control pueda volver al hilo principal en cualquier momento que se desee.

Para el hilo secundario se hace uso de una clase privada asociada a ConfiguracionFragmentActivity, que será tConectar, de tipo AsyncTask. Esta clase será la encargada de leer todos los datos de los *EditText* y componer la llamada de conexión a la base de datos, con un requisito impuesto, y es que por lo menos exista una dirección a la que intentar conectar. Si no se produce ningún error o excepción, se habrá conseguido establecer la conexión y se mostrará el mensaje de “Conectado Servidor MySQL”. Para

poder hacer la conexión a la base de datos, se tiene que añadir a las librerías del proyecto, el SDK de conexión a MariaDB, el archivo será ***mariadb-java-client-1.5.9.jar*** y para su correcto funcionamiento, se debe añadir también al gradle de la aplicación la siguiente línea

```
compile files('libs/mariadb-java-client-1.5.9.jar')
```

Por último, y debido a que es el primer momento en el que hacemos uso de una de las funciones de conexión a internet, debemos añadir al manifest de la aplicación el permiso:

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

Android considera este permiso de bajo riesgo, por lo que no hay que preocuparse de montar ningún sistema de verificación para saber si está activo o no. Directamente se le concede el permiso a la aplicación cuando quiere hacer uso de la red.

6.2.3 Clase Query

Debido a que a lo largo de toda la aplicación va a hacerse uso de la funcionalidad de conectar a la base de datos, se ha decidido implementar una clase independiente que sea llamada cada vez que se requiera una conexión. Esta clase será la clase Query.

La clase recibe en su constructor dos parámetros, un *String* llamado query, con la instrucción que se desea hacer sobre la base de datos y otra de tipo *Context* llamado context, que será el contexto de la actividad en la que se llama a esta clase. El contexto será muy importante porque en el propio constructor, será necesario su uso para poder cargar del archivo de preferencias de la aplicación, los datos de conexión a la base de datos:

```
SharedPreferences prefs = context.getSharedPreferences("BBDD", Context.MODE_PRIVATE);

IP = prefs.getString("Conexion", "");
contrasena = prefs.getString("Contraseña", "");
puerto = prefs.getInt("Puerto", 3307);
usuario = prefs.getString("Usuario", "");
```

Se dispone además de dos métodos principales, el primero será getResultado(), y cuya función principal es la de devolver un *ResultSet*, que es un tipo de datos que extiende de *Wrapper* y que está vinculado al SDK de MariaDB que hemos tenido que usar para poder conectar con la base de datos. Su función es la de devolvernos los resultados de la consulta en un formato empaquetado.

El método getResultado() ejecutará, al igual que pasaba en TabBBDD, un AsyncTask (con 30 segundos de limite) para poder conectar a la base de datos y por tanto se debe crear también una clase privada en la clase Query para tal acción.

La clase secundaria será BBDD, de tipo AsyncTask y realiza una función muy parecida a tConectar, pero es este caso además de conectar, manda una consulta a la base de datos y devuelve el resultado a la clase principal Query

```
String conexionMySQLURL = "jdbc:mariadb://" + IP + ":" + puerto
        + "/xgestevo?user=" + usuario + "&password=" + contrasena;

Class.forName("org.mariadb.jdbc.Driver");
Connection con = DriverManager.getConnection(conexionMySQLURL);
Statement st = con.createStatement();
ResultSet rs = st.executeQuery(query);
```

Tal y como puede verse en el trozo de código, se compone la conexión a la base de datos con los atributos que se trajeron de las preferencias del sistema y después, aplicando la api de conexión de MariaDB, se abre una conexión y se manda la consulta almacenada en query. Si todo es correcto, la base de datos da su respuesta en el *ResultSet* rs que será devuelto a Query y que a su vez se devolverá como resultado de getResultado().

El segundo método de la clase Query es getResultadoBlob() y que hace exactamente lo mismo que getResultado(), pero en esta ocasión hace uso de la clase privada BBDDBlob y cuya única diferencia con BBDD es la consulta y el resultado obtenido a la base de datos.

```
String conexionMySQLURL = "jdbc:mariadb://" + IP + ":" + puerto
        + "/xgestevo?user=" + usuario +"&password=" + contrasena;

Class.forName("org.mariadb.jdbc.Driver");
Connection con = DriverManager.getConnection(conexionMySQLURL);

PreparedStatement ps = con.prepareStatement(query);

//preparamos el Blob que vamos a insertar como un InputStream desde una ruta de archivo
File = new File(parametros[0]);
InputStream = new FileInputStream(file);
ps.setBlob(1,inputStream);
ps.setInt(2,(int) file.length());
//Ejecutamos la instrucción y obtenemos el número de resultados obtenidos
filas= ps.executeUpdate();
```

El motivo por el que se crea esta segunda función es porque la anterior BBDD se puede usar para consultar, modificar, crear o incluso borrar datos simples de la base de datos. Pero en algunos momentos de la aplicación será necesario el uso de datos más complejos de tipo Blob (en particular imágenes como ristra de Bytes). Por ese motivo se usa esta clase especial que compone una consulta que inserta los datos, de un archivo que pasamos como parámetro, como si fuera un dato de tipo Blob en la base de datos.

6.2.4 TabImpresora

Al igual que TabBBDD, esta clase es uno de los Fragment que carga una de las pestañas que componen la actividad ConfiguracionFragmentActivity. Es la que se encarga de la configuración de la conexión con la impresora Zebra con la que trabajará el terminal.

En este caso el layout asociado es, fragment_tab_impresora.xml. Contiene también una única función, además de las necesarias para la creación de la clase y la vista. Esta función es la de cargarConfiguracionImpresora() y permite, en este caso, obtener los parámetros almacenados para conectar a la impresora.

Devuelto el control de la aplicación a ConfiguracionFragmentActivity, hay también aquí dos métodos como en la Tab anterior, guardarConfiguracionImpresora(), con una función similar a la de guardarConfiguracionBBDD(), pero con los valores para la impresora y conectarImpresora() que es idéntica a la función conectar(), pero que hace uso de la clase privada tConectarImpresora. Esta clase en lugar de componer la conexión a la base de datos MySQL, lo que hace es abrir una conexión TCP con la impresora Zebra. Para poder hacer la conexión a la impresora, se tiene que añadir a las librerías del proyecto, el SDK de conexión a las impresoras Zebra, el archivo será ***ZSDK_ANDROID_API.jar*** y para su correcto funcionamiento, se debe añadir también al gradle de la aplicación la siguiente línea

```
compile files('libs/ZSDK_ANDROID_API.jar')
```

6.2.5 Clase Impresión

En esta ocasión también pasa lo mismo que con la clase Query, se va a usar la impresión en varias ocasiones y no resulta productivo reproducir el mismo código varias veces. Por tanto, se crea la clase Impresión para representar, de manera única, las conexiones a la impresora y la impresión con un formato predefinido. Recibe como parámetros el número de albarán a imprimir, la cantidad de etiquetas, el contexto para acceder a los datos del archivo de preferencias y a crear mensajes en la vista, y además recibe como parámetros dos objetos que son dos clases nuevas que se explicarán más adelante: PedidosAServir, con las cabeceras de los pedidos y Pedido, con los datos de las líneas servidas.

El método getResultado() establece la conexión con la impresora y devuelve una conexión abierta a la impresora en forma de objeto de la clase ZebraPrinter, para ello hace uso de la clase privada Enlace, que al igual que BBDD será una AsyncTask, y que establece una conexión con la impresora. Por su parte el método imprimirEtiquetas() recorre un bucle tantas veces como etiquetas queramos imprimir y cuya función es primero componer la etiqueta y después llamar a la función *write(etiqueta)* sobre la conexión abierta previamente con la impresora. El método desconectar() cierra definitivamente la conexión a la impresora.

Finalmente, el método generarEtiqueta() es el encargado de generar, como un Array de Bytes, la etiqueta que se le pasa a la impresora, para ello se obtiene la fecha actual del sistema, se obtienen los datos del destinatario, ya sea del objeto de la clase PedidosAServir, del cual se crea un objeto de la clase Cliente que también se verá un poco más adelante; o de lo contrario, si es un dropshipping, se saca de Xgestevo con la siguiente consulta:

```
String consultaDrop = "SELECT d.ENOM,d.EDOM,d.ECODPO,d.EPOB,d.ETEL" +
        " FROM fcdre001 d " +
        "where d.EPED=" + pedidos.getUnPedido();
```

A continuación, se le da al número de albarán un formato legible por el sistema de remesas de envío de Xgestevo. Debe tener 17 caracteres de la siguiente forma:

ALBXXXXYYYYZZZZZZ

Donde como ya se vio anteriormente X es la serie, Y el año y Z el albarán. Este último dato se añade, junto con los demás, a la etiqueta prediseñada con ZebraDesigner y se devuelve el Array resultante para mandarlo a la impresora.

6.3 Login

Esta actividad debe ser ejecutada siempre que se ejecute la aplicación, pues ninguna de las tres actividades restantes puede funcionar sin un usuario registrado. Por tanto, cualquiera de las otras tres actividades verifica si la variable global user está vacía y de ser así no se lanzará.

6.3.1 LoginActivity

Esta actividad es la que se lanza al pulsar sobre el botón de login de la MainActivity. Carga el layout activity_login.xml y su principal característica es que genera un *Spinner* que contendrá a todos los usuarios que pueden usarse para hacer login. Estos usuarios corresponden a todos los usuarios activos en Xgestevo. Esta carga de usuarios se hace gracias a la función getListadoUsuarios() de la clase Usuarios que se explica a un poco más adelante.

Al seleccionar un usuario del *Spinner*, se carga el almacén pre asignado a éste con la función numeroAlmacen() de la clase Usuarios. Este almacén puede cambiarse si se desea. Finalmente, tras pulsar el botón OK, se ejecuta el método login() de la clase Usuario, pasándole como parámetro el usuario seleccionado y la contraseña escrita en el EditText y que previamente se ha encriptado con el método encriptar(), que hace uso de mecanismo de encriptación que se explicó en la descripción del proyecto en el apartado 5.2 y que se solventa dividiendo la contraseña en caracteres CHAR y aplicando la siguiente instrucción:

```
c = (char) (pass.charAt(i) - 1 - i);
```

6.3.2 clase Usuarios

Esta clase sirve de apoyo a la actividad LoginActivity y su principal función es la de generar un vector de usuarios que tiene Xgestevo, para lo cual, realiza una conexión gracias a la clase Query y ejecuta la siguiente sentencia:

```
String consulta = "select xnombre,xcontra as Usuarios, xalmaini from fcusu order by xnombre";
```

Como puede observarse, por cada usuario se almacena, su nombre, contraseña encriptada y el almacén de inicio predeterminado. Además, se almacena el número total de usuarios que se han importado, o -1 si se produce algún error.

Los métodos de los que se dispone son getNumeroUsuarios() para obtener el total de usuarios; getListadoUsuarios() que devuelve un Array de nombres de usuario compuesto desde el vector del que se dispone; getNumeroAlmacen() que busca y devuelve el número de almacén de inicio del usuario pasado como parámetro; login() que, dados sus dos parámetros de usuario y contraseña, enfrenta esos valores con los que hay en el vector para encontrar una coincidencia exacta.

El vector está formado por elementos de una clase privada llamada Usuario. Cada objeto de esta clase contiene los tres elementos que se han traído de la base de datos de Xgestevo por cada usuario, su nombre de usuario, contraseña encriptada y almacén.

6.4 Consulta de artículos

6.4.1 ConsultaArticuloActivity

Esta actividad es la que se ejecuta tras pulsar el botón “Consultar Stock” de la MainActivity y verificar que hay un usuario registrado en el sistema. Carga el layout activity_consulta_articulo.xml. La principal función de esta actividad es la de cargar todos los elementos en la vista del articulo buscado, para lo cual se establecen dos opciones: pulsar el botón OK, con un escuchado en el botón; o pulsar la tecla enter del teclado, con un escuchador en el EditText esperando esa pulsación en particular. Ambos ejecutan el método consultarArticulo() y pasan como parámetro lo que haya escrito en el EditText.

El método consultarArtículo() se limita a crear un objeto de la clase Artículo y si el resultado es válido, carga todos los datos del artículo en los elementos de la vista.

Existe un botón “Modificar” que lanza la actividad ModificarArticuloActivity si se pulsa y si existe un artículo valido visualizándose actualmente, el artículo se manda como extra. Si recibe una modificación de esta actividad refresca los datos que se ven en la consulta.

6.4.2 Clase Articulo

Esta clase es la encargada de buscar los artículos en la base de datos de Xgestevo y devolver todos los datos que sean necesarios a cualquier otra clase que necesite hacer uso de ellos. Cada instancia de este objeto almacena los siguientes datos de un artículo valido: código, descripción, stand del almacén de consulta, categoría de rotación, peso, coste medio, stock del almacén de consulta, foto, si requiere número de serie y si está sujeto a inversión de sujeto pasivo.

Existen dos constructores de la clase, el primero busca en la base de datos la foto, y el segundo recibe la foto como parámetro, muy útil si se quiere que se le asigne otra foto o directamente no hace falta y se le pasa como Null para ahorrar recursos. En ambos casos, lo primero que se hace es lanzar la consulta a Xgestevo para localizar el artículo buscando la coincidencia del parámetro que se le pasa de código de artículo con cualquiera de sus posibles códigos en la base de datos:

```
String consulta = "select a.acodar, a.adescr, a.APESO, a.ACATROTAC, a.APMCOS, a.ARESSN3 " +
        "from fcart001 a " +
        "left join fcclt001 c on c.WCODAR=a.ACODAR " +
        "where \"" + codigo + "\" in ( a.ACODAR , a.ACODALT2 , a.ACODALT3 , a.ACODALT4 ," +
        "a.ACODALT5,a.ACODALT6,a.ACODALT7,a.ACODALT8 ,a.ACODALT9,a.ACODALT10,a.ARESCAR1) " +
        "or c.WCODALT=\"" + codigo + "\" limit 1";
```

En caso de que haya una coincidencia, se traen el resto de datos para completar todos los atributos de Articulo

```
consulta="select s.ASTOCK,s.ASTAND,f.FTFILE1,a.AINVSUJPAS,fam.FINVSUJPAS" +
        " from fcart001 a " +
        "inner join fcstk001 s on a.ACODAR=s.ACODAR and s.AALM="+almacen+
        " inner join fcfcp001 fam on fam.FCOD=a.ARESNUM4 " +
        "left join fcfos001 f on f.FTACODAR=a.ACODAR " +
        "where \"" + codArticulo + "\" = a.ACODAR " +
        "limit 1";
```

FTFILE1 y la línea del left join fcfos001 solo será necesaria si no se pasa como parámetro al constructor la foto. Si no existe foto, carga el recurso R.drawable.nofoto

El resto de métodos de métodos de la clase son modificarCaracteristica(), que ejecuta un update sobre cualquiera de los atributos básicos del artículo en las tablas fcart y fcstk. El método modificarImagen() hace un update sobre la tabla fcfos y hace uso del método getResultadoBlob() de la clase Query para poder actualizar la foto como un bloque de Bytes de tipo Blob.

El ultimo método es el de regularStock() que recibe como parámetro el stock inicial y lo compara con el stock almacenado actualmente en su atributo stock, si es distinto de 0 comienza con el proceso de actualización, para lo cual obtiene primero el número de cliente en el que tiene que hacer el albarán de regulación de stock

```
String consulta = "select GCLIREGSTK from fcge2001";
```

Por último, hace uso de las clases PedidosAServir y Pedido para generar la cabecera de un albarán en el cliente de regulación de stock y la línea de albarán de regulación con el artículo a regular y su diferencia de stock. El proceso de albarán se encarga de actualizar, como se verá más adelante, el stock del artículo.

6.4.3 ModificarArticuloActivity

Esta clase se lanza desde ConsultarArticuloActivity y recibe de esta como extra, el código de articulo a modificar. La actividad hace uso del layout activity_modificar_articulo.xml.

EL método consultarArticuloAModificar(), carga en la vista todos los datos del articulo recibido como extra de la actividad anterior.

Se establece un TextWatcher sobre cada EditText para verificar si hay algún cambio, así cuando se ejecute el método guardar() se actualicen o no los datos sobre el objeto Articulo.

El tratamiento de las modificaciones de la foto es un poco distinto, se establecen dos botones, uno para actualizar la foto tomando una instantánea desde la cámara, para lo cual hacemos uso de la actividad del sistema de Android que controla las capturas de la cámara:

```
Intent intent = new Intent("android.media.action.IMAGE_CAPTURE");
```

El otro método consiste en coger una foto de la galería, accediendo a la memoria del terminal, con otra actividad del sistema de Android:

```
Intent intent = new Intent(Intent.ACTION_PICK,
        android.provider.MediaStore.Images.Media.EXTERNAL_CONTENT_URI);
```

Según Android, el acceso a la memoria del terminal es un permiso peligroso y por tanto no se concede automáticamente sin aceptación. Por tanto, hace falta añadir al manifest de la aplicación el permiso de lectura en memoria:

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

Pero, por ser de alto riesgo, además deberá de controlarse la aceptación explicita del permiso, sobre todo si se trabaja con Android 6.0 o superior, a través del método onRequestPermissionsResult().

6.5 Realizar entradas

6.5.1 EntradaActivity

Esta es la actividad que se ejecuta tras seleccionar el botón “Realizar entrada” de la MainActivity. Carga el layout activty_entrada.xml con tres botones, cada uno de los cuales lanza la siguiente actividad SeleccionProveedorActivity, con un valor número distinto como extra para definir como se hará la entrada.

La actividad también crea una instancia de Entrada y PedidoProveedor que son tablas SQLite y se verán más adelante. Si hay datos en las tablas, dará a través de un

AlertDialog la opción de continuarla (pasa el parámetro 0 a la siguiente actividad) o cancelarla, con lo que borra todos los datos de las dos tablas.

6.5.2 SeleccionProveedorActivity

Esta actividad carga el layout activity_seleccion_proveedor y establece sobre el EditText en el que se pone el nombre del proveedor un *ArrayAdapter* que actuará como sistema de autocompletar. Este sistema se alimenta de un Array de String que es devuelto por el método buscarNombreProveedores() que recibe como parámetro el texto que se haya escrito en el EditText y busca semejanzas en la tabla de proveedores de Xgestevo:

```
String consulta= "select pnom from fcpro001 p where p.PNOM like \"%"+ nombre +"%\"";
```

Ya sea porque se elija un proveedor del sistema de autocompletar o porque se escriba su código, se crea un objeto de la clase proveedor para buscar si existe y pasar a la siguiente actividad de ServirEntradaActivity, pasando como parámetro el tipo de entrada elegido y el código de proveedor. Si el tipo de entrada es 0, es una recuperación y se omite esta actividad de selección de proveedor. Si es de tipo 3, es una entrada procesando pedidos de proveedor y deberá hacerse uso de la clase PedidoProveedor y del método cargarPedidosProveedor()

6.5.3 Clase Proveedor

Es una clase en la que cada objeto contiene los datos de un proveedor, con su código, su nombre y su forma de pago. Tiene dos constructores, en función de si se busca por nombre (con el autocompletar) o por código.

```
String consulta = "select p.PNOM, p.PFORPA from fcpro001 p where p.PCOD=" + codigo + " and
p.PBLOQUEADO=\"N\"";
String consulta = "select p.PCOD, p.PFORPA from fcpro001 p where p.PNOM=\"" + nombre + "\" and
p.PBLOQUEADO=\"N\"";
```

En ambos casos si el código termina siendo -1, es porque no se ha encontrado.

6.5.4 clase PedidoProveedor

Esta clase es para manejar una tabla en SQLite llamada pedidoProveedor, que contiene las líneas de pedido del proveedor pendientes de procesarse. Los datos que contiene por cada línea de la base de datos son: código de proveedor, numero de pedido, numero de línea, código del artículo, la cantidad pedida, su coste y la cantidad servida en este proceso. Para su carga inicial se usa el método cargarPedidosProveedor y que ejecuta la siguiente instrucción:

```
String consulta = "select l.LPED,l.LLINEA,l.LCODAR, l.LPRECI, l.LCANPEN," +
        "l.LALMACEN from fclop001 l " +
        "inner join fccop001 c on c.BPED=l.LPED and c.BLIQUID=\"N\" and c.BPED>1 " +
        "where l.LLIQUID=\"N\" and l.LPED>1  " +
        "and l.LALMACEN="+almacen+" and l.LCODPR="+ proveedor + " " +
        "order by l.LFECPED";
```

El método nuevaLinea() se cuándo se introduce un nuevo artículo en la entrada, comprueba si existe una coincidencia con cantidad pendiente en pedidoProveedor y si no es así, introduce una línea nueva no asociada a ningún pedido, de esa manera al finalizar el proceso, no descontará de ningún pedido pendiente, sino que se crea una cola.

modificarCantidadServida() es el método que se usa para modificar la cantidad de un artículo que se encuentra ya en el proceso de entrada. Actualiza la cantidad servida, comprobando si se van completando líneas y en caso de no quedar ninguna por completar, crea una nueva sin pedido como hace nuevaLinea()

El método actualizarPedido() es el método final que se usa al finalizar la actividad ServirEntradaActivity, recorre la tabla pedidoProvedor y actualiza las líneas de la tabla fclop de Xgestevo, sumando la cantidad servida a la que ya tiene. Si en alguna línea se encuentra que la cantidad pedida es igual a la servida, liquida la línea. Por último, revisa todas las líneas que ha actualizado y si todas las líneas pertenecientes a un mismo pedido están liquidadas, liquida también la cabecera en fccop.

6.5.5 ServirEntradaActivity

Esta actividad es la final de este proceso, y consiste en la introducción de los artículos que forman la entrada y su procesamiento según el tipo de entrada se haya seleccionado. La actividad hace uso del layout activity_servir_entrada cuya principal característica es el uso de un *RecyclerView* en el cual se va mostrando un listado de todos los artículos que se han introducido. Este *RecyclerView* hace uso de un adaptador personalizado que será la clase AdaptadorEntradas.

La principal función es la de insertarArticulosEntrada() que recibe como parámetro un código de artículo del EditText de la vista. Este código genera un objeto de tipo Articulo y si es válido, se inserta una línea nueva con la función insertarLinea() de la clase Entrada que se explica en el siguiente punto. Si la entrada es de tipo 3 también se usa el método nuevaLinea de la clase PedidoProveedor para descontar en cantidad 1 los artículos pendientes en los pedidos del proveedor.

Esta actividad también hace uso de un objeto *menu* entrada_menu que contendrá dos botones, uno para cancelar toda la operación y borra las tablas pedidoProveedor y entrada. Y el botón para procesar la entrada, que, según el tipo de entrada, llama a una función u otra de finalización en la clase Entrada.

Si la actividad ModificarLineaEntradaActivity devuelve un cambio sobre un artículo, se manda la orden al *RecyclerView* de refrescar la lista.

6.5.6 clase AdaptadorEntradas

Esta clase es una extensión de *RecyclerView.ViewHolder* y su principal función es la de crear cada una de las entradas del *RecyclerView* y mostrarlas con una vista personalizada en el layout linea_entrada.xml.

Cada línea es recuperada de la tabla entrada de la clase Entrada, a través la función getLinea(), que devuelve un objeto de la clase LineaEntrada de la posición solicitada.

6.5.7 clase LineaEntrada

Esta clase se usa para extraer los datos de cada línea de la tabla entrada. Es una clase muy simple y solo contiene dos atributos: el articulo y la cantidad que entra.

6.5.8 clase Entrada

Esta clase es para manejar una tabla en SQLite llamada entrada, que contiene los artículos que se están añadiendo en la actividad ServirEntradaActivity. Cada línea de la tabla contendrá el proveedor en el que se va a hacer la entrada, el artículo, la cantidad que entra, su coste, el tipo de entrada que se va a procesar, el almacén en el que entra el producto, el usuario que está usando el terminal y la forma de pago de pago del proveedor.

Sus principales métodos son el de insertarLinea() que inserta en la tabla una línea de articulo cada vez que se lee un código valido en el EditText de ServirEntradaActivity. borrarLinea(), que elimina una línea y reordena el _id de las demás para que el *RecyclerView* pueda seguir recuperándolas sin saltos y no de errores. También está modificarCantidadLinea() que modifica la cantidad del artículo que está en la línea que se pasa como parámetro al método. El método getLinea() que devuelve un objeto de tipo LineaEntrada usado por el *RecyclerView*.

Finalmente tenemos los tres métodos principales que finalizan el proceso de entrada según el tipo de entrada que se desee generar. El primero es generarCola() que coge cada línea de la tabla entrada y la inserta sus datos en una línea en la tabla de Xgestevo fcepd. Hace uso del método getSiguienteLineaEntradaCola() que coge el ultimo id incremental de fcepd y le suma 1. A todas las líneas les asigna el mismo número de cola, que se obtiene de igual manera que en la línea, con el método getSiguienteEntradaCola().

El segundo método es generarEntrada, que primeramente crea una cabecera de entrada en fccbe con los datos del proveedor y las cantidades totales de la tabla entrada, haciendo uso también de getSiguienteEntrada(), que hace algo parecido a lo que hace getSiguienteLineaEntradaCola(), pero en la tabla fccbe y aplicando un filtro para que tenga solo en cuenta lo elementos de la serie 0, que es en la que se trabaja. Después de generar la cabecera se recorre la tabla entrada y se insertan en la tabla de Xgestevo fclie cada una

de las líneas, asignándoles también un ordinal. Por cada línea que se inserta en fclie, se actualiza el stock y el precio medio de coste del artículo, para ello en fcart se suma el stock actual a la cantidad que entra y se calcula el coste medio mediante el cálculo proporcional entre el stock y coste actual y la cantidad que entra y su coste:

```
((a.ASTOCK*a.APMCOS)+("+cantidad+"*"+importe+"))/"+"(a.ASTOCK+"+cantidad+")
```

También se actualiza el stock del almacén de trabajo en fcstk de la misma forma.

El ultimo método es el de generarEntradaConPedido, que, al igual que el segundo método, genera una cabecera en fccbe y tantas líneas en fclie como líneas haya en este caso en pedidoProveedor con código de pedido mayor a 0, es decir, que vengan de un pedido que se encuentre pendiente en la tabla fclop. Estas líneas se obtienen con el método getLineasEntrada() de la clase PedidoProveedor. Otra vez se hace la misma actualización de stock por cada línea insertada en fclie. A continuación, se comprueba si quedan más elementos en la tabla pedidoProveedor y de ser así, se genera una cola, como en el primer modo de entrada, con los elementos restantes. Finalmente se llama al método actualizarPedidos() de PedidoProveedor para actualizar fclop y fccoc en Xgestevo.

6.5.9 ModificarLineaEntradaActivity

Esta actividad se ejecuta cuando se pulsa cualquier elemento del *RecyclerView* de ServirEntradaActivity y carga el layout activity_modificar_linea_entrada.xml. Su principal función es la de mostrar los datos del artículo que se ha seleccionado y permitir modificar la cantidad que entra. Para modificar la cantidad que entra hace uso del método modificarCantidadLinea() de la clase Entrada y en caso de ser de tipo 3 la entrada, también hace uso de modificarCantidadServida de la clase PedidoProveedor.

Esta actividad hace uso también de un recurso *menu* modificar_entrada_menu con el cual se da la opción de eliminar el registro en lugar de modificarlo. Para ello usa el método borrarLinea() de la clase Entrada y modificarCantidadServida de la clase PedidoProveedor.

6.6 Servir pedidos

6.6.1 PedidoActivity

Esta actividad se ejecuta al pulsar el botón “Servir Pedidos” de MainActivity. Carga el layout activty_pedido.xml con dos botones, cada uno de los cuales lanza la actividad SeleccionPedidosActivity, con un valor número distinto como extra para definir como se buscará el material para servir el pedido.

La actividad crea una instancia de Pedido y PedidoAServir que son tablas SQLite y se verán más adelante. Si hay datos en las tablas, mostrara un *AlertDialog* dando opción a continuar (pasa el parámetro 0 a la siguiente actividad) o cancelarla y vaciando las tablas.

6.6.2 SeleccionPedidosActivity

En esta actividad se van cargando los pedidos que en los siguientes pasos se procesarán. Hace uso del layout activity_seleccion_pedidos.xml y su comportamiento en el funcionamiento de la vista es muy similar al de ServirEntradaActivity, usa un *EditText* para recibir el número de pedido a analizar con el método insertarPedido() y si pasa todos los filtros, lo añade al listado que se ve en el *RecyclerView*, añade la cabecera de pedido a la tabla cabeceraPedido de la clase PedidosAServir y todas sus líneas a la tabla lineasPedido de la clase Pedido. Si no los pasa, devuelve un texto de error en función del fallo encontrado. Para el uso del *RecyclerView* se usa también un adaptador personalizado para mostrar el pedido, llamado AdaptadorPedidosAServir.

También se hace uso de un *menu* pedido_menu, que dará dos opciones, o borrar el pedido, que elimina todos los registros de la tabla cabeceraPedido de la clase PedidosAServir y de la tabla lineasPedido de la clase Pedido; o continuar sirviendo, que pasa a la actividad siguiente en función del tipo de pedido.

Un tipo de pedido 0 es una recuperación y no pasa por toda esta actividad, sino que se va directa a la actividad de servir pedido que corresponda al tipo de pedido que se está intentando servir.

En caso de pulsar sobre cualquiera de los elementos del *RecyclerView* se lanza la actividad ConsultaPedidoActivity que muestra todos los datos del pedido.

6.6.3 clase AdaptadorPedidosAServir

Esta clase es una extensión de *RecyclerView.ViewHolder* y muestra cada uno de los pedidos del *RecyclerView* con una vista personalizada en el layout cabecera_pedido.xml.

Cada línea es recuperada de la tabla cabeceraPedido de la clase PedidosAServir, a través la función getCabecera(), que devuelve un objeto de la clase CabeceraPedido de la posición solicitada.

6.6.4 clase CabeceraPedido

Esta clase se usa para extraer los datos de cada pedido de la tabla cabeceraPedido. Contiene los siguientes elementos de un pedido: código de cliente, centro de entrega, agencia de envío, almacén del pedido, forma de pago del pedido, número de artículos

pendientes del pedido, dirección de entrega de dropshipping, descuento del pedido, importe total de los artículos pendientes, numero de pedido.

6.6.5 ConsultaPedidoActivity

Es la actividad que se lanza cuando se pulsa sobre un elemento del *RecyclerView* de SeleccionPedidosActivity cargando el layout activity_consulta_pedido.xml, en el que se muestran los datos generales del pedido y un listado de los artículos pendientes que se deben servir, para ello se vuelve a usar un *ReciclerView* con un adaptador personalizado AdaptadorConsultaPedido.

Parte de los datos que se muestran son los datos del cliente, para lo que se usa la clase Cliente. Además, se muestran el total de artículos y de importe a servir que se obtienen de la tabla cabeceraPedido. Las líneas se obtienen de la tabla lineasPedido.

6.6.6 clase AdaptadorConsultaPedido

Clase extensión de *RecyclerView.ViewHolder* que muestra cada línea del pedido con un layout consulta_linea_pedido. Cada línea que usa el adaptador es recupera de la tabla lineasPedido con el método getLineaPedido() de la clase Pedido. En este caso se diferencia de los demás adaptadores porque se recupera, no por orden del campo _id de la tabla, sino por un cursor que se pasa como parámetro e indica cada vez que línea coger, que serán las del pedido seleccionado únicamente.

6.6.7 clase LineaPedido

Clase básica que se usa de apoyo con la tabla lineasPedido, con la que se representa cada línea de pedido. Tiene cuatro atributos: código de artículo, cantidad a servir, cantidad servida, importe.

6.6.8 clase Cliente

Esta clase es utilizada cada vez que se desea acceder a la base de datos de Xgestevo y recuperar los datos de un cliente. En su constructor se realiza esta consulta:

```
String consulta = "select c.CNOM, c.CINVSUJPAS, c.CDOM, c.CCODPO, c.CPOB, c.CTEL1 from
fccli001 c where c.CCODCL=" + codigo;
```

Sus atributos son: código de cliente, nombre de cliente, si el cliente está sujeto a inversión de sujeto pasivo, el domicilio del cliente, su código postal, su población y teléfono.

6.6.9 clase PedidoAServir

Esta clase representa una tabla SQLite llamada cabeceraPedido, en la que cada línea representa los datos generales de cada pedido que se pretende servir. Los campos que se almacenan de cada pedido son: el cliente al que pertenece, el número de pedido, dirección de dropshipping si la hubiera, el centro de destino, agencia por la que se envía, el

descuento que se le aplica, almacén sobre el que se va a servir, forma de pago aplicada, forma en la que se va a servir (directo o buscando), número de artículos, importe total, usuario que lo procesa. Antes de procesar un pedido, el código debe ser formateado, para lo cual se usa el método limpiarNumeroPedido() que, si es válido, lo devuelve como número.

El método insertarCabeceraPediso() inserta en la base de datos un pedido con la siguiente sentencia select:

```
String consulta = "select c.BCODCL, if(d.EDOM is null,\"\",d.EDOM), c.BCENTRO, c.BAGENCIA,"+
        " c.BDTO, c.BALMACEN, c.BFORPA," +
        " if(sum(l.LCANPEN) is null ,0,sum(l.LCANPEN)), " +
        " if (sum(LCANPEN*LPRECI) is null,0,sum(LCANPEN*LPRECI))" +
        " from fccoc001 c " +
        "left join fcdre001 d on d.EPED=c.BPED and d.EPED=" + pedido +
        "left join fcloc001 l on l.LPED=c.BPED and l.LCANPEN>0 and l.LLIQUID=\"N\" " +
        //revisar que la línea este reservada e impresas
        " and l.LRESSN1=\"S\" and l.LRESSN4=\"P\" " +
        " where c.BPED=" + pedido +
        " group by c.BPED" +
        //añadimos esta condición para que no nos devuelva ninguna fila tras hacer las sumas
        //en caso de no haber encontrado el pedido, pero que si que nos pueda devolver 0
        //en el caso de encontrar el pedido pero no tener filas pendientes
        " having c.BCODCL is not null";
```

Tras esta carga hace varios análisis y devuelve el número de pedidos encontrado o un código negativo en función del fallo encontrado.

Existen dos métodos para obtener el numero de la siguiente cabecera de albarán, y se calcula igual que la cabecera de entrada, pero en este caso se debe tener en cuenta si se aplica inversión de sujeto pasivo, con la serie 25, o no, con lo que se usa la serie 0.

El método generarCabeceraAlbaran inserta en la tabla fccba una línea por cada cabecera de albarán necesaria: 0 si no hay ISP, 25 si lo hay o ambas si es mixto el pedido. Además, si el campo dropshipping no está vacío, se deben recuperar todos los datos de la tabla fcdre de Xgestevo, pertenecientes a uno de los pedidos, y duplicarlos en una nueva línea vinculada al albarán.

Finalmente, el método insertarPedidoRegulacion() crea una cabecera perteneciente al cliente de regulación, solo con los datos de cantidad de articulo y su coste total.

6.6.10 clase Pedido

Esta clase también representa una base de datos SQLite, pero en esta ocasión de líneas de pedido en la tabla lineasPedido. Por cada línea almacena un ordinal, el cliente, el número de pedido, la línea del pedido, código del artículo, cantidad a servir, importe, descuento, cantidad servida, coste, si el articulo está sujeto a inversión de sujeto pasivo (o ISP), número de serie de los artículos, ubicación en el almacén, usuario que procesa el pedido. Esta tabla se llena a través del método insertarLineasPedido, que dado un numero de pedido, ejecuta la siguiente instrucción select sobre Xgestevo:

```
String consulta = "select l.LCODCL, l.LLINEA, l.LCODAR, l.LCANPEN, l.LPRECI, l.LDTO, " +
        "(l.LCOSTE/l.LCANTI) from fcloc001 l" +
        " where l.LPED=" + pedido +
        //solo lo reservado, impreso y pendiente
        " and l.LRESSN1=\"S\" and l.LRESSN4=\"P\" and l.LLIQUID=\"N\" and l.LCANPEN>0 ";
```

El método extraeCursorPedido() devuelve un cursor con el _id de todas las líneas de pedido que pertenecen a un pedido y es usado por la clase AdaptadorConsultaPedido.

El método asignarNumeroSerie() concatena un nuevo número de serie al *String* ya existente de número de serie de un artículo en la tabla de lineasPedido.

Existen en esta clase varios métodos que en función de si el pedido aplica o no ISP usa unos u otros. Puede verse en getImporteServido(), getCosteServido() y getPeso() en los que hay tres métodos por cada uno: el total, solo con líneas de ISP o solo sin ISP.

Lo anterior dicho también se aplica sobre el método generarLineas(), que inserta en la tabla fclia de Xgestevo una línea por cada línea en la que se ha servido mínimo un artículo de la tabla lineasPedido. Al igual que pasaba en las entradas, cada línea en fclia modifica el stock, pero en esta ocasión de manera negativa en vez de positiva, pero el tipo de operativa es la misma, actualizar fcstk y fcart, aunque sin recalcular el coste medio en esta ocasión. generarLineas() inserta todas las líneas generarLineasISP() solo las de ISP y generarLineasNoISP() las que no se le aplica ISP.

El método actualizarPedidos() recorre todos los pedidos de la tabla fcloc de Xgestevo de los cuales se ha generado una línea en fclia, cosa que se sabe gracias a que se almacena el número de pedido y la línea a la que pertenece el artículo. Si la cantidad servida es igual a la solicitada se liquida la línea. Tras revisar todas las líneas, se revisan igualmente las cabeceras y si todas sus líneas están liquidadas, se liquida el pedido.

Finalmente, el método insertarLineasRegulacion() genera una línea, asociada al cliente de regulación de stock, con el código de artículo a regular y la cantidad a regular.

6.6.11 ServirPedidoBuscandoActivity

Esta actividad es ejecutada cuando el tipo de pedido a servir es 1 y carga el layout activity_servir_pedido_buscando.xml. Lo primero que hace es ejecutar el método ordenarLineas() de la clase Pedido y buscar el primer artículo ordenado con el método siguienteArticulo() que obtiene el siguiente articulo con cantidad pendiente de servir. A continuación, se usa el método cargarArticulo() que muestra todos los datos en la vista.

Cada vez que se pulsa el botón OK o se pulsa la tecla enter del teclado, se ejecuta el método insertarArticuloPedido, que crea un objeto de la clase Articulo y comprueba si es igual al que se busca. De ser así, usa el método incrementarServirOrdenado de la clase

Pedido, que suma 1 a la cantidad servida de la línea que está en pantalla. Además, comprueba si es necesario número de serie en el artículo y de ser así, lanza un *AlertDialog* con un *EditText* solicitándolo. Después hace uso del método asignarNumeroSerie() para almacenar el contenido de dicho EditText. Si la cantidad servida es la misma que la solicitada, automáticamente se vuelve a ejecutar los métodos siguienteArticulo() y cargarArticulo() para ir al próximo de la lista, a no ser que ya no quede nada por servir y por tanto se da por terminado el proceso y se deshabilita el botón siguiente.

Existen dos botones que terminan el proceso, el de cancelar que elimina los datos de las dos tablas y vuelve al principio de la aplicación; y el de terminar, que procesa los pedidos y genera los albaranes con el método terminarPedido(). Este método verifica si la empresa está sujeta a ISJ, al igual que si el cliente también lo está y en función de esos resultados va llamando a los métodos necesarios para crear las cabeceras de albarán y las líneas que pertenecen a cada uno de ellos, según también los artículos que se han servidor y ejecuta el método de actualizarPedidos() para cerrar el proceso con Xgestevo. Finalmente, se hace uso de la clase Impresión y de su constructor para imprimir tantas etiquetas como se haya indicado en el AlertView que apareció al pulsar sobre el botón terminar.

6.6.12 ServirPedidoDirectoActivity

Esta actividad es ejecutada cuando el tipo de pedido a servir es 2 y carga el layout activity_servir_pedido_buscando.xml. Esta actividad es muy similar a la anterior, pero en lugar de mostrar un solo artículo, lo que hace es usar un *RecyclerView* para mostrar todas las líneas de pedido que se deben servir. Para ello hace uso del adaptador personalizado AdaptadorPedidos. Hace uso del método insertarArticuloPedido(), que realiza lo mismo que en la actividad anterior, pero esta vez se limita a incrementar la cantidad servida del articulo y no necesita buscar el siguiente, pues están todos visibles en el *RecyclerView*. Los métodos cancelarPedido() y terminarPedido() son los mismos métodos que se usan en la actividad ServirPedidoBuscandoActivity.

6.6.13 clase AdaptadorPedidos

Ultima clase extensión de *RecyclerView.ViewHolder. Se vuelve a mostrar, otra vez, los datos de una línea de pedido, pero esta vez con layout línea_pedido.xml y en esta ocasión, la forma de presentarlos es ordenados por ubicación, gracias al método getLineaPedidoOrdenado() de la clase Pedido. Devuelve un objeto de la clase LineaPedido que ya se vio anteriormente, pero se ha recorrido la tabla lineasPedido no por _id, sino por un campo orden que se ha aplicado previamente.*

7. Evaluación

El proceso de pruebas ha sido continuo a lo largo de todo el proyecto y ha consistido en la realización de pruebas de caja blanca para comprobar el funcionamiento de cada una de las funciones y clases que se iban programando; pruebas de caja negra para comprobar el estado final de cada una de las funcionalidades de los módulos, según se iban acabando; y pruebas con usuarios finales que han aportado sus impresiones y fallos al proyecto.

Para comprobar el cumplimiento de los requisitos del proyecto y, tal y como estaba planeado en el modelo de desarrollo incremental que se vio en el apartado 4.3, se han desarrollado prototipos funcionales a la finalización de cada ciclo iterativo. Todos los prototipos han sido testados en entorno de pruebas y posteriormente en un entorno real por personal de almacén acostumbrado a trabajar con Regumobile de Xgestevo. Dicho entorno real ha sido el almacén del mayorista de informática Depau Sistemas, S.L., en Cartagena.

7.1 Dispositivos para las pruebas

Los prototipos, tanto en pruebas como en entorno real, han sido instalados en tres tipos de terminales móviles, conectados a base de datos local de Xgestevo:

Tabla 4. Terminales móviles para pruebas

	WIKO Jerry Max	SAMSUNG J3	WIKO Pulp
Procesador	Quad-core 1,3 GHz	Quad-core 1,5 GHz	Octa-core 1,4 GHz
Memoria RAM	1 GB	1,5 GB	2 GB
Pantalla	5"	5"	5"
Versión Android	Anroid 6.0 Marshmallow	Android 5.1 Lollipop	Android 5.1 Lollipop
Batería	4900 mAh	2600 mAh	2500 mAh
Peso	186g	138g	149g

La introducción de datos se ha realizado tanto de manera manual como a través de un lector de códigos de barras Bluetooth de la marca Honeywell, modelo 1602G2d.

La impresión de etiquetas se ha hecho en una impresora térmica, con conexión a red local, de la marca Zebra y modelo GK420D, usando etiquetas de tamaño 100x80mm.

7.2 Pruebas sobre entorno real

A continuación, se expondrán las pruebas realizadas en el entorno real, seguido de algunas mejoras que se han implementado en algunos apartados como consecuencia.

7.2.1 Pruebas de configuración

En la configuración se ha probado inicialmente la conexión a la base de datos y se ha podido verificar una rápida respuesta de la aplicación a cualquier error tanto en usuario,

como en contraseña y puerto de conexión. En cambio, en cuanto la dirección del servidor, se producían bloqueos al dejar en blanco el campo y se ha filtrado para no permitir esa situación. Aunque se permite la cancelación del proceso de conexión, el hilo de tConectar sigue en ejecución intentando conectar si el servidor de destino está realizando otra tarea y no responde, por lo que no puede realizar ninguna otra conexión, de ahí que se implementen los 30 segundos de ejecución máxima del hilo.

En cambio, las pruebas de impresora no han dado ningún fallo que deba ser corregido, si los datos son erróneos o la impresora está ocupada, informa inmediatamente.

7.2.2 Pruebas de login

Cuando la base datos no está configurada o no disponible, responde rápidamente con un aviso de volver atrás y configurarla. Además, la carga del spinner se produce prácticamente instantánea, si la respuesta del servidor es rápida. No se aprecian fallos en el proceso de verificación de usuario y contraseña.

7.2.3 Pruebas de consulta de artículos

Se verifica que, ni en esta ni en las siguientes dos secciones se puede acceder si una identificación correcta. En la ventana de consultas se aprecia un ligero retraso en la carga de la información, dependiendo del tamaño de la foto almacenada en Xgestevo. Además, resulta incómodo el que aparezca continuamente el teclado, por lo que se implementa, en todos los apartados de la aplicación en los que haya la posibilidad de una introducción de texto a través lector, la ocultación del teclado. La modificación de datos del artículo y del stock se trasladan de manera correcta a Xgestevo, incluido el caso de la captura de fotos o la búsqueda en la galería, que de manera efectiva maneja el permiso.

7.2.4 Pruebas de realizar entrada

El proceso de selección de proveedor y el autocompletar resultan rápidos y válidos. En el caso de una recuperación, lleva al pedido directamente al proceso de visualizar el material ya introducido. Pero, en este proceso, como en el de la vuelta tras modificar algún artículo, se producen grandes retrasos si hay muchos artículos en la vista, pues de cada uno tiene que cargar su foto y datos. La finalización de la entrada resulta correcta en cualquiera de sus modalidades, pero lenta si hay muchos artículos.

7.2.5 Pruebas de servir pedidos

Al igual que en el caso anterior se aprecian ciertos retrasos en grandes cargas de datos, como una recuperación, o la carga de un pedido muy grande, pero son bastante menores al no cargarse su foto. El proceso de finalización es correcto y actualiza tanto pedidos, como albaranes y stock de artículos.

8. Conclusiones y trabajo futuro

8.1 Conclusiones

El proyecto tenía como finalidad el crear una aplicación que sirviera para realizar las operaciones comunes en un almacén que tuviera como sistema de gestión al software Xgestevo. Se han estudiado todas las alternativas que existen en la actualidad para realizar este trabajo y se ha demostrado la necesidad de una nueva alternativa, más moderna y con más funcionalidades que se adapte al concepto de un almacén logístico moderno.

Desde el comienzo del proceso de análisis y codificación se ha podido comprobar que, el modelo de desarrollo elegido se ha adaptado perfectamente a la estructuración del software por módulos. El comenzar por los módulos de configuración, centrándose solo en la conexión y el posterior encapsulamiento de esta funcionalidad en clases independientes y accesibles por toda la aplicación, ha sido la base para todo el sistema de comunicaciones con Xgestevo. El módulo de login da permiso a usar las tres funcionalidades principales. Por último, el de consulta de artículos es la base para todo el manejo de entradas y albaranes, que ya son dos actividades finales y totalmente independientes.

Partiendo de los requisitos funcionales, el resultado final de XAlmacen satisface todos y cada uno de ellos. Los tres módulos de trabajo principales que son el de consulta y modificación de stock, las entradas de almacén y el sistema para servir pedidos, han cumplido sus objetivos, son totalmente funcionales y los resultados introducidos en la base de datos de Xgestevo son válidos y coherentes.

Las pruebas, aunque satisfactorias en todos sus resultados, han demostrado que algunos de los procesos son demasiado lentos y en ocasiones confusos, porque la aplicación se queda a la espera del resultado por parte de Xgestevo y parece en ocasiones estar bloqueada, aunque realmente está trabajando y comunicándose con la base de datos.

Los requisitos de apariencia simple, funcional y con los colores de fondo de Xgestevo, termina resultando, según la experiencia de los usuarios de las pruebas reales, algo monótona y en ocasiones de difícil lectura en las partes en las que el azul del fondo es más oscuro.

También las pruebas han demostrado que, aunque es más agradable y en ocasiones, más útil, el uso de las fotos en el proceso de entrada, enlentece demasiado el trabajo y deberá ser revisado, mejorado o suprimido.

Aun, con estos inconvenientes, XAlmacen ha comenzado a usarse, en su versión actual, por el personal del almacén en el que se han realizado las pruebas. Se prevén muchas mejoras, revisiones y el incremento en el personal dedicado al proyecto, junto con la incorporación de gran número de funcionalidades nuevas.

8.2 Trabajo futuro

Dadas las conclusiones alcanzadas a la finalización del proyecto, se establecen a corto plazo las siguientes revisiones y mejoras:

- Revisión del proceso de generación de líneas de entrada y de albarán para tratar de acelerar el proceso de finalización.
- Generación de imágenes en miniatura de las que almacena Xgestevo, para acelerar la carga de imágenes en XAlmacen.
- Revisar estilos y apariencia de la aplicación para conseguir evitar la sensación de monotonía y las zonas oscuras del final de la pantalla.

A medio plazo se plantean las siguientes mejoras:

- Poder usar distintas series en la aplicación además de la 0
- Poder usar más empresas además de la principal 001.
- Adaptar la aplicación al sistema de almacenamiento caótico que actualmente se está desarrollando en Xgestevo.

A largo plazo se establecen los siguientes objetivos en la aplicación:

- Añadir al sistema de servir pedidos, un sistema de ubicación de artículos a través de pick-to-line, un sistema por el cual una luz le indica al usuario la ubicación exacta del próximo artículo que debe coger.
- Sistema de preparación de pedidos por olas. En el cual el usuario del programa puede preparar varios pedidos de clientes distintos en los cuales debe ir a recoger material semejante que se encuentre en las mismas zonas. Optimización del recorrido del personal del almacén. Finalizando en un sistema de put-to-light en el que una luz le indica por cada artículo, la caja en la que lo debe meter.

Referencias y enlaces

Anaya Tejero, J.J. (2011). Almacenes: Análisis, diseño y organización. Madrid: ESIC Editorial.

Ansgar Becker. (2017). HeidiSQL. Junio 29, 2017, de Ansgar Becker Recuperado de: https://www.heidisql.com/

dBase. (2017). dBase. Abril 22, 2017, de dBase, LLC. Recuperado de https://en.wikipedia.org/wiki/DBase

EFE. (2008). Sun Microsystems compra MySQL por 1.000 millones de dólares. Abril 17, 2017, de Ediciones EL PAIS SL. Recuperado de http://tecnologia.elpais.com/tecnologia/2008/01/16/actualidad/1200475685_850215.html

Elgin, B. (2005). Google Buys Android for Its Mobile Arsenal. Abril 12, 2017, de Bloomberg Businessweek. Recuperado de https://web.archive.org/web/20110205190729/http://www.businessweek.com/technology/content/aug2005/tc20050817_0949_tc024.htm

EP. (2009). Oracle adquiere Sun Microsystems por 5.710 millones. Abril 17, 2017, de Ediciones EL PAIS SL. Recuperado de http://tecnologia.elpais.com/tecnologia/2009/04/20/actualidad/1240216080_850215.html

Gartner. (2009). Gartner Says Worldwide Smartphone Sales Reached Its Lowest Growth Rate With 3.7 Per Cent Increase in Fourth Quarter of 2008. Abril 15, 2017, de Gartner Inc. Recuperado de http://www.gartner.com/newsroom/id/910112

Gartner. Mayo (2010). Gartner Says Worldwide Mobile Phone Sales Grew 17 Per Cent in First Quarter 2010. Abril 15, 2017, de Gartner Inc. Recuperado de http://www.gartner.com/newsroom/id/1372013

Gartner. Agosto (2010). Gartner Says Worldwide Mobile Device Sales Grew 13.8 Percent in Second Quarter of 2010, But Competition Drove Prices Down. Abril 15, 2017, de Gartner Inc. Recuperado de http://www.gartner.com/newsroom/id/1421013

Gartner. Noviembre (2010). Gartner Says Worldwide Mobile Phone Sales Grew 35 Percent in Third Quarter 2010; Smartphone Sales Increased 96 Percent. Abril 15, 2017, de Gartner Inc. Recuperado de http://www.gartner.com/newsroom/id/1466313

Gartner. (2011). Gartner Says Worldwide Mobile Device Sales to End Users Reached 1.6 Billion Units in 2010; Smartphone Sales Grew 72 Percent in 2010. Abril 15, 2017, de Gartner Inc. Recuperado de http://www.gartner.com/newsroom/id/1543014

Gartner. Mayo (2012). Gartner Says Worldwide Sales of Mobile Phones Declined 2 Percent in First Quarter of 2012; Previous Year-over-Year Decline Occurred in Second Quarter of 2009. Abril 15, 2017, de Gartner Inc. Recuperado de http://www.gartner.com/newsroom/id/2017015

Gartner. Agosto (2012). Gartner Says Worldwide Sales of Mobile Phones Declined 2.3 Percent in Second Quarter of 2012. Abril 15, 2017, de Gartner Inc. Recuperado de http://www.gartner.com/newsroom/id/2120015

Gartner. Noviembre (2012). Gartner Says Worldwide Sales of Mobile Phones Declined 3 Percent in Third Quarter of 2012; Smartphone Sales Increased 47 Percent. Abril 15, 2017, de Gartner Inc. Recuperado de http://www.gartner.com/newsroom/id/2237315

Gartner. Febrero (2013). Gartner Says Worldwide Mobile Phone Sales Declined 1.7 Percent in 2012. Abril 15, 2017, de Gartner Inc. Recuperado de http://www.gartner.com/newsroom/id/2335616

Gartner. Mayo (2013). Gartner Says Asia/Pacific Led Worldwide Mobile Phone Sales to Growth in First Quarter of 2013. Abril 15, 2017, de Gartner Inc. Recuperado de http://www.gartner.com/newsroom/id/2482816

Gartner. Agosto (2013). Gartner Says Smartphone Sales Grew 46.5 Percent in Second Quarter of 2013 and Exceeded Feature Phone Sales for First Time. Abril 15, 2017, de Gartner Inc. Recuperado de http://www.gartner.com/newsroom/id/2573415

Gartner. Diciembre (2014). Gartner Says Sales of Smartphones Grew 20 Percent in Third Quarter of 2014. Abril 15, 2017, de Gartner Inc. Recuperado de http://www.gartner.com/newsroom/id/2944819

Gartner. Marzo (2015). Gartner Says Smartphone Sales Surpassed One Billion Units in 2014. Abril 15, 2017, de Gartner Inc. Recuperado de http://www.gartner.com/newsroom/id/2996817

Gartner. Mayo (2015). Gartner Says Emerging Markets Drove Worldwide Smartphone Sales to 19 Percent Growth in First Quarter of 2015. Abril 15, 2017, de Gartner Inc. Recuperado de http://www.gartner.com/newsroom/id/3061917

Gartner. Agosto (2015). Gartner Says Worldwide Smartphone Sales Recorded Slowest Growth Rate Since 2013. Abril 15, 2017, de Gartner Inc. Recuperado de http://www.gartner.com/newsroom/id/3115517

Gartner. Mayo (2016). Gartner Says Worldwide Smartphone Sales Grew 3.9 Percent in First Quarter of 2016. Abril 15, 2017, de Gartner Inc. Recuperado de http://www.gartner.com/newsroom/id/3323017

Gartner. Agosto (2016). Gartner Says Five of Top 10 Worldwide Mobile Phone Vendors Increased Sales in Second Quarter of 2016. Abril 15, 2017, de Gartner Inc. Recuperado de http://www.gartner.com/newsroom/id/3415117

Gartner. Noviembre (2016). Gartner Says Chinese Smartphone Vendors Were Only Vendors in the Global Top Five to Increase Sales in the Third Quarter of 2016. Abril 15, 2017, de Gartner Inc. Recuperado de http://www.gartner.com/newsroom/id/3516317

Gartner. (2017). Gartner Says Worldwide Sales of Smartphones Grew 7 Percent in the Fourth Quarter of 2016. Abril 15, 2017, de Gartner Inc. Recuperado de http://www.gartner.com/newsroom/id/3609817

Git. (2017). GutHub. Junio 29, 2017, de Software Freedom Conservancy Recuperado de: https://git-scm.com/downloads

Google. Android (2017). La historia de Android. Abril 14, 2017, de Google Inc. Recuperado de https://www.android.com/intl/es_es/history/

Google. Android developer, compatibilidad (2017). Funciones de la biblioteca de compatibilidad. Abril 14, 2017, de Google Inc. Recuperado de https://developer.android.com/topic/libraries/support-library/features.html?hl=es-419

Google. Android developer, plataforma (2017). Arquitectura de la plataforma. Abril 15, 2017, de Google Inc. Recuperado de https://developer.android.com/guide/platform/index.html?hl=es

Google. Android developer, versiones. (2017). Versiones de la plataforma. Junio 29, 2017, de Google Inc. Recuperado de: https://developer.android.com/about/dashboards/index.html?hl=es-419

Google. Android Studio. (2017). Conoce Android Studio. Junio 29, 2017, de Google Inc. Recuperado de: https://developer.android.com/studio/intro/index.html

HTC. (2008). T-Mobile Unveils the T-Mobile G1 — the First Phone Powered by Android. Abril 13, 2017, de HTC Corp. Recuperado de http://web.archive.org/web/20081111051233/http://www.htc.com/www/press.aspx?id=66338

IBM. (2006). DB2 Version 9.1 for Linux, UNIX and Windows manuals. Abril 22, 2017, de IBM. Recuperado de http://www-01.ibm.com/support/docview.wss?uid=swg27009552

ISO. (2014). ISO/IEC 14882:2014. Abril 22, 2017, de International Organization for Standardization. Recuperado de https://www.iso.org/standard/64029.html

ISO. (2011). ISO/IEC 9899:2011. Abril 22, 2017, de International Organization for Standardization. Recuperado de https://www.iso.org/standard/57853.html

Java. (2017). ¿Qué es Java? Abril 22, 2017, de Oracle Corporation. Recuperado de https://www.java.com/es/about/whatis_java.jsp

JetBrains s.r.o. (2017). IntelliJ IDEA: the Java IDE for Professional Developers. Junio 29, 2017, de JetBrains s.r.o. Recuperado de: https://www.jetbrains.com/idea/

Laravel. (2017). Laravel Manuals. Abril 22, 2017, de Laravel. Recuperado de https://laravel.com/docs/5.4

Linux. (2016). What is Linux? Abril 22, 2017, de The Linux Foundation. Recuperado de https://www.linux.com/what-is-linux

MariaDB. (2016). Knowledge Base. Abril 22, 2017, de MariaDB Recuperado de https://mariadb.com/kb/en/

Microsoft. (2005). ASCII Character Codes. Abril 24, 2017, de Microsoft Corp. Recuperado de https://msdn.microsoft.com/en-us/library/4z4t9ed1(v=vs.80).aspx

Microsoft Corporation. (1994). Developer´s guide Microsoft FoxPro. U.S.A.: Microsoft Press.

Microsoft Corporation. (1998). Microsoft Visual FoxPro: Programmer's Guide. U.S.A.: Microsoft Press.

MySQL. (2017). MySQL Connector/J 5.1 Developer Guide. Abril 23, 2017, de Oracle. Recuperado de https://dev.mysql.com/doc/connector-j/5.1/en/

MySQL News. (1998). Version News about MySQL. Abril 17, 2017, de TCX DataKonsult AB. Recuperado de http://web.archive.org/web/19981207025553/http://www.tcx.se:80/news.html

Open Handset Alliance. (2007). Open Handset Alliance Releases Android SDK. Abril 12, 2017, de Open Handset Alliance. Recuperado de https://www.openhandsetalliance.com/press_111207.html

Open Handset Alliance. (2008). Google and the Open Handset Alliance Announce Android Open Source Availability. Abril 13, 2017, de Open Handset Alliance. Recuperado de https://www.openhandsetalliance.com/press_102108.html

Portaltic/EP. (2011). Android supera a Symbian como líder del mercado de 'smartphones'. Abril 14, 2017, de Agencia Europa Press. Recuperado de http://www.europapress.es/portaltic/software/noticia-android-supera-symbian-lider-mercado-smartphones-20110131164159.html

Reardon, M. (2008). Google Android prototypes debut at MWC. Abril 12, 2017, de CNET Interactive Inc. Recuperado de https://www.cnet.com/news/google-android-prototypes-debut-at-mwc/

Saura, D. (2015). Quiénes somos. Junio 19, 2017, de Software de Gestión Xgest, S.L. Recuperado de: http://xgestevo.net/web2015/index.php

SourceForge. (1999). Find, Create, and Publish Open Source software for free. Junio 29, 2017, de VA Software Recuperado de: https://sourceforge.net/

Spence, R. (1993). Rick Spence's Clipper 5.2: Power Programmer's Guide/Book and Disk. U.S.A: Sybex Inc.

Tomás, J. (2017). El gran libro de Android. España: Marcambo, S.A.

Visual Studio .NET. (2003). FoxISAPI Automation Server Samples. Abril 22, 2017, de Microsoft Corp. Recuperado de https://msdn.microsoft.com/es-es/library/aa979317(v=vs.71).aspx

Zebra. (2016). Zebra Programming Guide. Abril 26, 2017, de Zebra Technologies Corporation. Recuperado de https://www.zebra.com/content/dam/zebra/manuals/en-us/software/zpl-zbi2-pm-en.pdf

Zebra. (2017). SOFTWARE BÁSICO DE ETIQUETAS DE CÓDIGOS DE BARRAS ZEBRADESIGNER. Junio 29, 2017, de ZIH Corp Recuperado de: https://www.zebra.com/la/es/products/software/barcode-printers/zebralink/zebra-designer.html

Anexo1. Tablas de Xgestevo

Tabla de terminales móviles aaaa_pdadatXXX

Esta tabla contiene los datos de comunicación entre los terminales móviles y el servidor de movilidad de Xgestevo. No es usada en el desarrollo de este proyecto, pero, es importante conocer su funcionamiento para conocer cómo funciona el actual Regumobile.

La tabla está compuesta por 6 campos, cuya descripción es la siguiente:

Tabla 5. Campos de la tabla aaaa_pdadatXXX

Campo	Tipo	Null	Key	Default	Descripción
npda	int(4)	SI	UNI	0	Id único de la línea
idpda	varchar(20)	SI			Identificador de la PDA (número de serie)
estado	int(2)	SI		0	0 Comunicación terminada/Respuesta del servidor 1 mensaje enviado al servidor 2 mensaje recogido por el servidor
data	mediumtext	NO		NULL	Bloque de datos transmitidos
idserver	varchar(20)	SI			Servidor al que va destinado el mensaje
dtime	datetime	NO		01/01/2000 0:00	Fecha y hora de última modificación

En la tabla hay una entrada por cada terminal, identificada de manera única por un ID auto incremental npda y por el número de serie de la terminal en idpda.

El funcionamiento básico consiste en que el terminal hace una solicitud añadiendo código XML en data y poniendo el estado a 1 e identificando al servidor al que hace la solicitud en idserver.

El servidor recoge la solicitud borrando el dato y el idserver y poniendo el estado a 2. Procesa los datos que el terminal le ha solicitado y los devuelve en data y poniendo el estado a 0 tanto para informar al terminal que ya tiene los datos, como diciendo que está listo para la siguiente solicitud.

Tabla de artículos fcartXXX

Esta es la tabla en la que se encuentran la gran mayoría de los datos del artículo. La tabla está compuesta por un total de 206 campos, de los cuales solo veremos los más importantes.

Tabla 6. Campos de la tabla fcartXXX

Campo	Tipo	Null	Key	Default	Descripción
ACODAR	char(28)	SI	UNI		Código del articulo
adescr	char(150)	NO	MUL		Descripción del artículo
APVP1	double	SI		0	Precio de venta 1
APVP2	double	SI		0	Precio de venta 2
APVP3	double	SI		0	Precio de venta 3
APVP4	double	SI		0	Precio de venta 4
ATIPIVA	int(11)	SI	MUL	2	Tipo de IVA que se le aplica, 1 0% 2 21% 3 4% 4 10%
ASTOCK	double	SI	MUL	0	Stock general de todos los almacenes
APRCOS	double	SI		0	Último coste de entrada
APMCOS	double	SI		0	Coste medio de entrada
ASTAND	char(20)	SI			Stand general del almacén en el que se ubica
APESO	double	SI		0	Peso
AOBSE	mediumtext	NO		NULL	Observaciones del articulo
ABLOQUEADO	char(1)	SI	MUL	N	Articulo bloqueado (S/N)
ANOINVENT	char(1)	SI		N	Artículo no inventariable (S/N)
APREBASE	double	SI		0	Precio de base para el cálculo de los precios de venta
ARESCAR1	char(30)	SI	MUL		Código alternativo 1 - Part Number
AFAMILIA	int(11)	SI	MUL	0	Familia de compras
ARESNUM4	double	SI	MUL	0	Familia de ventas
ARESNUM5	double	SI		0	Precio de venta 5
ARESNUM6	double	SI		0	Precio de venta 6
ACODALT2	char(30)	SI	MUL		Código alternativo 2
ACODALT3	char(30)	SI	MUL		Código alternativo 3
ACODALT4	char(30)	SI	MUL		Código alternativo 4
AAMPDES	mediumtext	NO		NULL	Ampliación de descripción
XUSUARIO	char(20)	SI			Usuario que lo dio de alta
XFECALTA	date	SI		01/01/1980	Fecha de alta
ACODALT5	char(30)	SI	MUL		Código alternativo 5
ACODALT6	char(30)	SI	MUL		Código alternativo 6
ACODALT7	char(30)	SI	MUL		Código alternativo 7
ACODALT8	char(30)	SI	MUL		Código alternativo 8

ACODALT9	char(30)	SI	MUL		Código alternativo 9
ACODALT10	char(30)	SI	MUL		Código alternativo 10
AMARCA	char(30)	SI	MUL		Marca del articulo
ALARGO	double	SI		1	Largo
AANCHO	double	SI		1	Ancho
AALTO	double	SI		1	Alto
AINVSUJPAS	char(1)	SI		N	Articulo con Inversión de Sujeto Pasivo
AEMBALABLE	char(1)	SI		N	Embalable (S/N)
ACATROTAC	char(10)	SI		C	Categoría de rotación

Tabla de cabeceras de albaranes fccbaXXX

En esta tabla se almacena uno de los dos elementos de los que consta un albarán, su cabecera. Esta cabecera tendrá los datos generales del albarán, como los datos del cliente, los datos de envío y los totales entre otros. La tabla consta de 62 campos, de los cuales, los más importantes son los siguientes.

Tabla 7. Campos de la tabla fccbaXXX

Campo	Tipo	Null	Key	Default	Descripción
BALBA	double(14,0)	SI	UNI	0	Número de albarán
BALMACEN	int(11)	SI	MUL	0	Almacén de las piezas del albarán
BFACT	char(1)	SI	MUL	N	Facturado (S/N)
BCODCL	int(11)	SI	MUL	0	Código de cliente
BFECHA	date	SI	MUL	01/01/1980	Fecha del albarán
BTOBRU	double	SI		0	Total bruto
BTOCOS	double	SI		0	Total coste
BFACTURA	double(14,0)	SI	MUL	0	Número de factura
ACTUALIZAD	char(1)	SI			Actualizado (S/N)
BFORPA	int(11)	SI	MUL	0	Forma de pago
BOBSE	mediumtext	NO		NULL	Observaciones
BUSUARIO	char(20)	SI	MUL		Usuario que crea el albarán
BHORAALB	char(8)	SI			Hora de creación del albarán
BDTO	double	SI		0	Descuento general
BCENTRO	int(11)	SI	MUL	0	Centro de entrega de material
BAGENCIA	int(11)	SI	MUL	0	Agencia de transporte
BBULTOS	double	SI		1	Bultos del albarán
BPESO	double	SI		0	Peso del albarán
BOBSINT	mediumtext	NO		NULL	Observaciones internas
BULTMODIF	timestamp	NO		NULL	Fecha y hora de la modificación
BRESFEC1	date	SI	MUL	01/01/1980	Fecha última modificación
BUSUMODIF	char(20)	SI	MUL		Último usuario que la modifica

Tabla de cabeceras de entrada fccbeXXX

En esta tabla se almacena uno de los dos elementos de los que consta una entrada, su cabecera. Esta cabecera tendrá los datos generales de la entrada, como los datos del proveedor y los costes totales. La tabla consta de 39 campos, de los cuales, los más importantes son los siguientes.

Tabla 8. Campos de la tabla fccbeXXX

Campo	Tipo	Null	Key	Default	Descripción
BENT	double(14,0)	SI	UNI	0	Número de entrada
BALMACEN	int(11)	SI	MUL	0	Almacén de la entrada
BCODPR	int(11)	SI	MUL	0	Código de proveedor
BFECHA	date	SI	MUL	01/01/1980	Fecha de creación de la entrada
BTOBRU	double	SI		0	Total coste
BOBSE	mediumtext	NO		NULL	Observaciones
BUSUARIO	char(20)	SI	MUL		Usuario que crea la entrada
ACTUALIZAD	char(1)	SI			Actualizado (S/N)
BHORAENT	char(8)	SI			Hora de la entrada
BDTO	double	SI		0	Descuento de la entrada
BRESFEC1	date	SI	MUL	01/01/1980	Fecha última modificación
BUSUMODIF	char(20)	SI	MUL		Último usuario que la modifica
BOBSINT	mediumtext	NO		NULL	Observaciones internas
BULTMODIF	timestamp	NO		NULL	fecha y hora de la modificación

Tabla de centros de trabajo de clientes fccenXXX

En esta tabla se almacenan los datos relacionados con los distintos centros de trabajo o puntos de entrega de material que puede tener un cliente. La tabla consta de 37 campos, de los cuales, los más importantes son los siguientes.

Tabla 9. Campos de la tabla fccenXXX

Campo	Tipo	Null	Key	Default	Descripción
ZCLI	int(11)	SI	MUL	0	Código de cliente
ZCEN	int(11)	SI	MUL	0	Código de centro del cliente
ZNOM	char(50)	SI	MUL		Nombre del centro
ZDOM	char(50)	SI			Dirección del centro
ZCODPO	char(8)	SI	MUL		Código postal del centro
ZPOB	char(40)	SI	MUL		Población del centro
ZPAIS	char(20)	SI			Provincia del centro
ZTEL	char(20)	SI			Teléfono del centro
ZFAX	char(20)	SI			Fax del centro
ZDESACT	char(1)	SI		N	Desactivado (S/N)
ZOBS	mediumtext	NO		NULL	Observaciones del centro
ZMAIL1	char(52)	SI			Correo del centro
ZCODIGOUNI	int(11)	NO	PRI	NULL	Id auto incremental
ZTARI	int(11)	SI		0	Tarifa del centro

Tabla de clientes fcclixxx

En esta tabla se almacenan los principales relacionados con los clientes, sus datos personales y fiscales entre otros. La tabla consta de 213 campos, de los cuales, los más importantes son.

Tabla 10. Campos de la tabla fccliXXX

Campo	Tipo	Null	Key	Default	Descripción
CCODCL	int(11)	SI	UNI	0	Código de cliente
CNOM	char(50)	SI	MUL		Nombre
CDOM	char(50)	SI			Dirección
CCODPO	char(8)	SI	MUL		Código postal
CPOB	char(40)	SI	MUL		Población
CPAIS	char(20)	SI			Provincia
CTEL1	char(20)	SI			Teléfono
CFAX1	char(20)	SI			Fax
CMAIL1	char(52)	SI			Correo electrónico
CDNI	char(20)	SI	MUL		DNI
CFORPA	int(11)	SI		1	Forma de pago
CBANCO	char(30)	SI			Banco
CENT	char(4)	SI			Entidad
CSUC	char(4)	SI			Sucursal
CDIG	char(2)	SI			Digito de control
CCUE	char(10)	SI			Número de cuenta
CTARI	int(11)	SI		3	Tarifa del cliente
CBLOQUEADO	char(1)	SI		N	Bloqueado (S/N)
COBS	mediumtext	NO		NULL	Observaciones
XUSUARIO	char(20)	SI			Usuario que lo dio de alta
XFECALTA	date	SI		01/01/1980	Fecha de alta
CAGENCIA	int(11)	SI	MUL	0	Agencia de envió por defecto
CCENDEFEC	int(11)	SI	MUL	0	Centro por defecto
CINVSUJPAS	char(1)	SI		N	Sujeto a Inversión de sujeto pasivo

Tabla de códigos alternativos fccltXXX

En esta tabla se almacenan los códigos alternativos que no pueden introducirse en la tabla de artículos fcartxxx porque el artículo en cuestión tenga demasiados códigos. La tabla está compuesta por un total de 12 campos, de los cuales los siguientes son los más importantes y usados.

Tabla 11. Campos de la tabla fccltXXX

Campo	Tipo	Null	Key	Default	Descripción
WCODAR	char(28)	SI	MUL		Código de articulo principal
WCODALT	char(40)	SI	MUL		Código alternativo
WCANTI	double	SI		1	Cantidad que se sirve con este código
WULTMODIF	timestamp	NO		NULL	Fecha de última modificación
WCODIGOUNI	int(11)	NO	PRI	NULL	Identificador único por cada línea

Tabla de cabeceras de ofertas y pedidos de clientes fccocxxx

En esta tabla se almacenan todos los datos relacionados con las ofertas y pedidos de clientes, como pueden ser los datos del cliente y dirección de destino, datos de facturación e importes totales. La tabla está compuesta por un total de 61 campos, de los cuales los siguientes son los más importantes y usados.

Tabla 12. Campos de la tabla fccocXXX

Campo	Tipo	Null	Key	Default	Descripción
BOFE	double(14,0)	SI	UNI	0	Número de oferta
BPED	double(14,0)	SI	MUL	0	Número de pedido
BESPED	char(1)	SI	MUL	N	Indica si la oferta ha pasado a pedido (S/N)
BCODCL	int(11)	SI	MUL	0	Código de cliente
BFECOFE	date	SI	MUL	01/01/1980	Fecha de la oferta
BFECPED	date	SI	MUL	01/01/1980	Fecha del pedido
BTOBRU	double	SI		0	Total bruto
BTOCOS	double	SI		0	Total coste
BOBSE	mediumtext	NO		NULL	Observaciones
BUSUARIO	char(20)	SI	MUL		Usuario que crea el documento
BLIQUID	char(1)	SI	MUL	N	Pedido liquidado (S/N)
ACTUALIZAD	char(1)	SI			Documento actualizado (S/N)
FECACTU	date	SI		01/01/1980	Fecha de actualización
BHORAOFE	char(8)	SI			Hora de creación de la oferta
BHORAPED	char(8)	SI			Hora de creación del pedido
BDTO	double	SI		0	Descuento general
BALMACEN	int(11)	SI		0	Almacén del pedido
BUSUMODIF	char(20)	SI	MUL		Usuario que modifica el documento
BCENTRO	int(11)	SI	MUL	0	Centro de envió del cliente
BOBSINT	mediumtext	NO		NULL	Observaciones internas
BFORPA	int(11)	SI	MUL	0	Forma de pago
BAGENCIA	int(11)	SI	MUL	0	Agencia de envió
BULTMODIF	timestamp	NO		NULL	Fecha última modificación

Tabla de cabeceras de ofertas y pedidos de proveedor fccopxxx

En esta tabla se almacenan todos los datos relacionados con las ofertas y pedidos de proveedores, como pueden ser los datos del proveedor, datos de facturación e importes totales. La tabla está compuesta por un total de 52 campos, de los cuales los siguientes son los más importantes y usados.

Tabla 13. Campos de la tabla fccopXXX

Campo	Tipo	Null	Key	Default	Descripción
BOFE	double(14,0)	SI	UNI	0	Número de oferta
BPED	double(14,0)	SI	MUL	0	Número de pedido
BESPED	char(1)	SI		N	Indica si la oferta ha pasado a pedido (S/N)
BCODPRO	int(11)	SI	MUL	0	Código de proveedor
BFECOFE	date	SI		01/01/1980	Fecha de creación de la oferta
BFECPED	date	SI		01/01/1980	Fecha de creación del pedido
BPLAZO	date	SI		01/01/1980	Plazo de entrega
BTOBRU	double	SI		0	Total Coste
BOBSE	mediumtext	NO		NULL	Observaciones
BOBSINT	mediumtext	NO		NULL	Observaciones internas
BUSUARIO	char(20)	SI	MUL		Usuario que crea el documento
BLIQUID	char(1)	SI	MUL	N	Indica si está el pedido liquidado (S/N)
ACTUALIZAD	char(1)	SI			Documento actualizado (S/N)
FECACTU	date	SI		01/01/1980	Fecha de actualización
BHORAOFE	char(8)	SI			Hora de creación de la oferta
BHORAPED	char(8)	SI			Hora de creación del pedido
BDTO	double	SI		0	Descuento general
BALMACEN	int(11)	SI	MUL	0	Almacén del pedido
BUSUMODIF	char(20)	SI	MUL		Usuario que modifica el documento
BFORPA	int(11)	SI	MUL	0	Forma de pago
BULTMODIF	timestamp	NO		NULL	Fecha de la última modificación

Tabla de direcciones de envió de clientes fcdrexxx

En esta tabla se almacenan las direcciones de envió que el cliente proporciona para que el material de un pedido sea mandado a un tercero ajeno a la relación comercial. Es la tabla de dropshipping. La tabla consta de 20 campos, de los cuales los más usados y de mayor importancia son los siguientes.

Tabla 14. Campos de la tabla fcdreXXX

Campo	Tipo	Null	Key	Default	Descripción
ECOD	double(14,0)	SI	UNI	0	Id único de cada registro
EPED	double(14,0)	SI	MUL	0	Número del pedido asociado
EALB	double(14,0)	SI	MUL	0	Número del albarán asociado
ENOM	char(50)	SI			Nombre del tercero
EDOM	char(50)	SI			Dirección del tercero
ECODPO	char(8)	SI			Código postal del tercero
EPOB	char(40)	SI			Población del tercero
EPAIS	char(20)	SI			Provincia del tercero
ETEL	char(20)	SI			Teléfono del tercero
EFAX	char(20)	SI			Fax del tercero
EOBS	mediumtext	NO		NULL	Observación del envió
ECODCLI	int(11)	SI	MUL	0	Código del cliente
EREEMB	double	SI		0	Valor de contra reembolso del pedido
EULTMODIF	timestamp	NO		NULL	Fecha de la última modificación

Tabla de colas de entrada de proveedor fcepdxxx

En esta tabla se almacenan una a una todas las líneas que pertenecen a una cola de entrada de proveedor, sobre todo almacena que artículos y qué cantidad han llegado al almacén. La tabla consta de 14 campos, de los cuales, los más importantes son los siguientes.

Tabla 15. Campos de la tabla fcepdXXX

Campo	Tipo	Null	Key	Default	Descripción
RCODIGO	double(14,0)	SI	UNI	0	Id único de la línea
RENTRADA	double(14,0)	SI	MUL	0	Número de cola a la que pertenece la línea
RLINEA	int(11)	SI	MUL	0	Ordinal asignado a la línea dentro de la cola
RFECHA	date	SI	MUL	01/01/1980	Fecha de creación de la línea
RHORA	char(8)	SI			Hora de creación de la línea
RUSUARIO	char(20)	SI			Usuario que ha creado la línea
RCODPRO	int(11)	SI	MUL	0	Código de proveedor
RCODAR	char(28)	SI	MUL		Código de articulo
RCANTI	double	SI		0	Cantidad del articulo
RCANASIG	double	SI		0	Cantidad del articulo asignada a una entrada
RLIQUID	char(1)	SI	MUL	N	Línea liquidada (S/N)
RALM	int(11)	SI	MUL	0	Almacén en la que entra el material de la cola

Tabla de facturas de cliente fcfacxxx

En esta tabla se guardan todas las facturas de cliente que se van generando. En sus campos se guardan todos los datos necesarios para la contabilización del movimiento del almacén que la ha generado. La tabla consta de 100 campos, de los cuales, los más usado e importantes son los siguientes.

Tabla 16. Campos de la tabla fcfacXXX

Campo	Tipo	Null	Key	Default	Descripción
FDOC	double(14,0)	SI	UNI	0	Número de factura
FFECHA	date	SI	MUL	01/01/1980	Fecha de la factura
FCODCL	int(11)	SI	MUL	0	Código del cliente
FTOBRU	double	SI		0	Total bruto
FDTO	double	SI		0	Descuento total
FIMPDTO	double	SI		0	Importe del descuento
FGASFI	double	SI		0	Gastos financieros
FBAS1	double	SI		0	Base con tipo de IVA 1
FBAS2	double	SI		0	Base con tipo de IVA 2
FBAS3	double	SI		0	Base con tipo de IVA 3
FBAS4	double	SI		0	Base con tipo de IVA 4
FBAS5	double	SI		0	Base con tipo de IVA 5
FIVA1	double	SI		0	Valor del IVA 1
FIVA2	double	SI		0	Valor del IVA 2
FIVA3	double	SI		0	Valor del IVA 3
FIVA4	double	SI		0	Valor del IVA 4
FIVA5	double	SI		0	Valor del IVA 5
FREC1	double	SI		0	Recargo de equivalencia del tipo de IVA 1
FREC2	double	SI		0	Recargo de equivalencia del tipo de IVA 2
FREC3	double	SI		0	Recargo de equivalencia del tipo de IVA 3
FREC4	double	SI		0	Recargo de equivalencia del tipo de IVA 4
FREC5	double	SI		0	Recargo de equivalencia del tipo de IVA 5
FBASTOT	double	SI		0	Base total
FIVATOT	double	SI		0	IVA total
FRECTOT	double	SI		0	Recargo de equivalencia total
FTOTAL	double	SI		0	Total de la factura
FFORPA	int(11)	SI		0	Forma de pago
FCOSTE	double	SI		0	Coste total
FENLA	char(1)	SI	MUL	N	Enlazada en contabilidad (S/N)
FMODIF	char(1)	SI	MUL	N	Modificada (S/N)
FASI	double(10,0)	SI	MUL	0	Asiento contable
ACTUALIZAD	char(1)	SI			Actualizada (S/N)
FECACTU	date	SI		01/01/1980	Fecha de actualización

FNOMCL	char(50)	SI			Nombre del cliente
FDOMCL	char(50)	SI			Dirección del cliente
FCPCL	char(8)	SI			Código postal del cliente
FPOBCL	char(40)	SI			Población del cliente
FCIFCL	char(20)	SI			Cif o DNI del cliente
FPAIS	char(20)	SI			Provincia del cliente
FOBSE	mediumtext	NO		NULL	Observaciones
FUSUARIO	char(20)	SI	MUL		Usuario que crea la factura
FUSUMODIF	char(20)	SI	MUL		Usuario que modifica la factura
FFECHORCRE	datetime	SI	MUL	01/01/1980 0:00	Fecha de creación
FFECHORENL	datetime	SI	MUL	01/01/1980 0:00	Fecha de enlace con contabilidad
FIMPRESA	char(1)	SI		N	Impresa (S/N)
FULTMODIF	timestamp	NO		NULL	Fecha de modificación
FRECTIFI	char(1)	SI		N	Rectificativa (S/N)
FINVSUJPAS	char(1)	SI		N	Inversión de sujeto pasivo (S/N)

Tabla de familia de ventas de productos fcfcpXXX

En esta tabla se guardan todas las familias por las cuales los productos se clasifican para su venta. La tabla consta de 35 campos, de los cuales, los más usado e importantes son los siguientes.

Tabla 17. Campos de la tabla fcfcpXXX

Campo	Tipo	Null	Key	Default	Descripción
FCOD	int(11)	SI	UNI	0	Código de familia de ventas
FDES	char(50)	SI	MUL		Descripción de la familia de ventas
FOBS	mediumtext	NO		NULL	Observaciones
FGRUPO	int(11)	SI	MUL	0	Grupo de familia de ventas superior
XUSUARIO	char(20)	SI			Usuario que crea la familia
XFECALTA	date	SI		01/01/1980	Fecha de creación
FULTMODIF	timestamp	NO		NULL	Fecha de última modificación
FINVSUJPAS	char(1)	SI		N	Familia sujeta a Inversión de sujeto pasivo

Tabla de fotos de artículos fcfosxxx

En esta tabla se almacenan los bloques de datos que corresponden a los archivos multimedia relacionados con cada artículo, normalmente fotografías. Tiene un total de 17 campos, y estos son sus campos principales.

Tabla 18. Campos de la tabla fcfosXXX

Campo	Tipo	Null	Key	Default	Descripción
FTCOD	double(15,0)	SI	UNI	0	Id único de la línea
FTART	char(50)	SI	MUL		Nombre del fichero
FTFECHA	datetime	SI		01/01/1980 0:00	Fecha de creación del registro
FTFILE1	mediumblob	SI		NULL	Datos de la foto 1
FTSIZE1	double(14,0)	SI	MUL	0	Tamaño de la foto 1
FTFILE2	mediumblob	SI		NULL	Datos de la foto 2
FTSIZE2	double(14,0)	SI	MUL	0	Tamaño de la foto 2
FTFILE3	mediumblob	SI		NULL	Datos de la foto 3
FTSIZE3	double(14,0)	SI	MUL	0	Tamaño de la foto 3
FTFILE4	mediumblob	SI		NULL	Datos de la foto 4
FTSIZE4	double(14,0)	SI	MUL	0	Tamaño de la foto 4
FTFILE5	mediumblob	SI		NULL	Datos de la foto 5
FTSIZE5	double(14,0)	SI	MUL	0	Tamaño de la foto 5
FTFILETMB	mediumblob	SI		NULL	Datos de la foto miniatura
FTSIZETMB	double(14,0)	SI	MUL	0	Tamaño de la foto miniatura
FTACODAR	char(28)	SI	MUL		Código de articulo

Tabla de facturas de proveedor fcfprxxx

En esta tabla se guardan todas las facturas de proveedor que se van generando. En sus campos se guardan todos los datos necesarios para la contabilización del movimiento del almacén que la ha generado. La tabla consta de 119 campos, de los cuales, los más usado e importantes son los siguientes.

Tabla 19. Campos de la tabla fcfprXXX

Campo	Tipo	Null	Key	Default	Descripción
FDOC	double(14,0)	SI	UNI	0	Número de factura
FFACTURA	char(20)	SI	MUL		Número de factura ofrecida por el proveedor
FFECHA	date	SI	MUL	01/01/1980	Fecha de factura
FCODPR	int(11)	SI	MUL	0	Código de proveedor
FTOBRU	double	SI		0	Coste total
FDTO	double	SI		0	Descuento
FIMPDTO	double	SI		0	Importe total del descuento
FGASFI	double	SI		0	Total gastos financieros
FBAS1	double	SI		0	Base con tipo de IVA 1
FBAS2	double	SI		0	Base con tipo de IVA 2
FBAS3	double	SI		0	Base con tipo de IVA 3
FBAS4	double	SI		0	Base con tipo de IVA 4
FBAS5	double	SI		0	Base con tipo de IVA 5
FIVA1	double	SI		0	Valor del IVA 1
FIVA2	double	SI		0	Valor del IVA 2
FIVA3	double	SI		0	Valor del IVA 3
FIVA4	double	SI		0	Valor del IVA 4
FIVA5	double	SI		0	Valor del IVA 5
FREC1	double	SI		0	Recargo de equivalencia del tipo de IVA 1
FREC2	double	SI		0	Recargo de equivalencia del tipo de IVA 2
FREC3	double	SI		0	Recargo de equivalencia del tipo de IVA 3
FREC4	double	SI		0	Recargo de equivalencia del tipo de IVA 4
FREC5	double	SI		0	Recargo de equivalencia del tipo de IVA 5
FBASTOT	double	SI		0	Base total
FIVATOT	double	SI		0	IVA total
FRECTOT	double	SI		0	Recargo de equivalencia total
FTOTAL	double	SI		0	Total de la factura
FFORPA	int(11)	SI		0	Forma de pago
FCOSTE	double	SI		0	Coste total
FENLA	char(1)	SI	MUL	N	Enlace con contabilidad (S/N)
FMODIF	char(1)	SI	MUL	N	Modificada (S/N)
FASI	double(10,0)	SI	MUL	0	Asiento contable
ACTUALIZAD	char(1)	SI			Actualizada (S/N)
FECACTU	date	SI		01/01/1980	Fecha de actualización
FOBSE	mediumtext	NO		NULL	Observaciones
FUSUARIO	char(20)	SI	MUL		Usuario que crea la factura

FUSUMODIF	char(20)	SI	MUL		Usuario que modifica la factura
FINVSUJPAS	char(1)	SI		N	Factura sujeta a Inversión de sujeto pasivo

Tablas de datos generales de la empresa fcgenXXX fcge2XXX y fcge3XXX

En estas tres tablas se guardan todos los datos de configuración de cada empresa. Está dividido en tres tablas pues la cantidad de campos que hay en cada tabla es tan grande que puede llegar a ser inmanejable.

La tabla fcgenXXX tiene un total de 253 campos.

Tabla 20. Campos de la tabla fcgenXXX

Campo	Tipo	Null	Key	Default	Descripción
GCODEMP	char(3)	SI	UNI		Código de empresa
GEMPRES	char(50)	SI			Nombre de la empresa
GCIF	char(15)	SI			Cif de la empresa
GALMACEN	int(11)	SI		1	Almacén principal
GEJERC	int(11)	SI		0	Ejercicio contable actual
GIVA1	double	SI		0	% de IVA 1
GIVA2	double	SI		0	% de IVA 2
GIVA3	double	SI		0	% de IVA 3
GIVA4	double	SI		0	% de IVA 4
GIVA5	double	SI		0	% de IVA 5
GREC1	double	SI		0	% de recargo de equivalencia 1
GREC2	double	SI		0	% de recargo de equivalencia 2
GREC3	double	SI		0	% de recargo de equivalencia 3
GREC4	double	SI		0	% de recargo de equivalencia 4
GREC5	double	SI		0	% de recargo de equivalencia 5

La tabla fcge2XXX tiene 253 campos también.

Tabla 21. Campos de la tabla fcge2XXX

Campo	Tipo	Null	Key	Default	Descripción
GCODEMP	char(3)	SI	UNI		Código de empresa
GCLIREGSTK	int(11)	SI		0	Código del cliente que se usa para crear albaranes de regulación de stock

La tabla fcge3XXX tiene 158 campos.

Tabla 22. Campos de la tabla fcge3XXX

Campo	Tipo	Null	Key	Default	Descripción
GCODEMP	char(3)	SI	UNI		Código de empresa
GEMPSUJPAS	char(1)	SI		N	Empresa sujeta a Inversión de sujeto pasivo (S/N)

Tabla de líneas de albarán fcliaXXX

Es esta tabla se almacena el segundo elemento perteneciente a los albaranes, las líneas de artículos que lo componen. Por cada línea podemos ver que articulo y su cantidad, junto con los datos que relacionan a la línea con el pedido, albarán y cliente. La tabla consta de un total de 52 campos, de los cuales los más usados e importantes son.

Tabla 23. Campos de la tabla fcliaXXX

Campo	Tipo	Null	Key	Default	Descripción
LALBA	double(14,0)	SI	MUL	0	Número de albarán
LLINEA	int(11)	SI	MUL	0	Ordinal de la línea
LALMACEN	int(11)	SI	MUL	0	Almacén del albarán
LCODCL	int(11)	SI	MUL	0	Código de cliente
LFECHA	date	SI	MUL	01/01/1980	Fecha del albarán
LCODAR	char(28)	SI	MUL		Código de articulo
LCANTI	double	SI		0	Cantidad
LPRECI	double	SI		0	Precio de venta
LDTO	double	SI		0	Descuento por articulo
LIMPOR	double	SI		0	Importe total cantidad*precio
LCOSTE	double	SI		0	Coste total
LNUMPED	double(14,0)	SI	MUL	0	Número de pedido de origen
LLINPED	int(11)	SI	MUL	0	Ordinal de la línea de pedido
LULTMODIF	timestamp	NO		NULL	Fecha última modificación

Tabla de líneas de entrada fclieXXX

Es esta tabla se almacena el segundo elemento perteneciente a las entradas, las líneas de artículos que lo componen. Por cada línea podemos ver que articulo y su cantidad, junto con los datos que relacionan a la línea con la entrada y proveedor. La tabla consta de un total de 55 campos, de los cuales los más usados e importantes son.

Tabla 24. Campos de la tabla fclieXXX

Campo	Tipo	Null	Key	Default	Descripción
LENT	double(14,0)	SI	MUL	0	Número de entrada
LLINEA	int(11)	SI	MUL	0	Ordinal de la línea
LALMACEN	int(11)	SI	MUL	0	Almacén de la entrada
LFECHA	date	SI	MUL	01/01/1980	Fecha de la entrada
LCODAR	char(28)	SI	MUL		Código de articulo
LCODPR	int(11)	SI	MUL	0	Código de proveedor
LCANTI	double	SI	MUL	0	Cantidad
LPRECI	double	SI		0	Precio de compra
LDTO	double	SI		0	Descuento
LIMPOR	double	SI		0	Importe total cantidad*precio
LULTMODIF	timestamp	NO		NULL	Fecha última modificación

Tabla de líneas de ofertas y pedidos de cliente fclocXXX

Es esta tabla se almacena el segundo elemento perteneciente a las ofertas y pedidos de clientes, las líneas de artículos que lo componen. Por cada línea podemos ver que articulo y su cantidad, junto con los datos que relacionan a la línea con la oferta, pedido y cliente. La tabla consta de un total de 47 campos, de los cuales los más usados e importantes son.

Tabla 25. Campos de la tabla fclocXXX

Campo	Tipo	Null	Key	Default	Descripción
LOFE	double(14,0)	SI	MUL	0	Número de oferta de cliente
LPED	double(14,0)	SI	MUL	0	Número de pedido de cliente
LLINEA	int(11)	SI	MUL	0	Ordinal de la línea
LFECOFE	date	SI	MUL	01/01/1980	Fecha de oferta
LFECPED	date	SI	MUL	01/01/1980	Fecha de pedido
LCODAR	char(28)	SI	MUL		Código de articulo
LCODCL	int(11)	SI	MUL	0	Código de cliente
LCANTI	double	SI	MUL	0	Cantidad
LPRECI	double	SI		0	Precio unitario
LDTO	double	SI		0	Descuento
LIMPOR	double	SI		0	Importe total
LCANSER	double	SI	MUL	0	Cantidad servida
LCANPEN	double	SI		0	Cantidad pendiente
LRESSN1	char(1)	SI		N	Línea reservada (S/N)
LRESSN4	char(1)	SI	MUL	N	Línea impresa (P/N)
LLIQUID	char(1)	SI	MUL	N	Línea liquidada (S/N)
LCOSTE	double	SI		0	Coste total
LALMACEN	int(11)	SI	MUL	0	Almacén de la línea
LULTMODIF	timestamp	NO		NULL	Fecha última modificación

Tabla de líneas de ofertas y pedidos de proveedor fclopXXX

Es esta tabla se almacena el segundo elemento perteneciente a las ofertas y pedidos de proveedor, las líneas de artículos que lo componen. Por cada línea podemos ver que articulo y su cantidad, junto con los datos que relacionan a la línea con la oferta, pedido y proveedor. La tabla consta de un total de 42 campos, de los cuales los más usados e importantes son.

Tabla 26. Campos de la tabla fclopXXX

Campo	Tipo	Null	Key	Default	Descripción
LOFE	double(14,0)	SI	MUL	0	Número de oferta de proveedor
LPED	double(14,0)	SI	MUL	0	Número de pedido de proveedor
LLINEA	int(11)	SI	MUL	0	Ordinal de la línea
LFECOFE	date	SI	MUL	01/01/1980	Fecha de oferta
LFECPED	date	SI	MUL	01/01/1980	Fecha de pedido
LCODAR	char(28)	SI	MUL		Código de articulo
LCODPR	int(11)	SI	MUL	0	Código de proveedor
LCANTI	double	SI		0	Cantidad
LPRECI	double	SI		0	Precio unitario
LDTO	double	SI		0	Descuento
LIMPOR	double	SI		0	Importe total
LCANPEN	double	SI	MUL	0	Cantidad pendiente
LLIQUID	char(1)	SI	MUL	N	Línea liquidada (S/N)
LALMACEN	int(11)	SI	MUL	0	Almacén de la entrada
LULTMODIF	timestamp	NO		NULL	Fecha de la última modificación

Tabla de números de serie fcnsrXXX

Es esta tabla se almacena los números de serie de los artículos que entran y salen del almacén. Por cada Por cada artículo en el que se guarde un número de serie al hacer una entrada o un albarán, se creará una línea en esta tabla. La tabla consta de un total de 11 campos, de los cuales los más usados e importantes son.

Tabla 27. Campos de la tabla fcnsrXXX

Campo	Tipo	Null	Key	Default	Descripción
NCODAR	char(28)	YES	MUL		Código del articulo
NNUMSER	char(50)	YES	MUL		Número de serie
NFECHA	date	YES	MUL	01/01/1980	Fecha de inserción del número de serie
NTIPMOV	char(1)	YES	MUL		Tipo de movimiento: E-->Entra al almacén S-->Sale del almacén
NTIPDOC	char(1)	YES	MUL		Tipo de documento: E-->Entrada de proveedor A-->Albarán de cliente
NDOC	double(14,0)	YES	MUL	0	Número de documento
NLINEA	int(11)	YES	MUL	0	Número de línea
NALMA	int(11)	YES	MUL	0	Número de almacén

Tabla de PDF o XML de facturas fcpdfXXX

En esta tabla se almacenan como bloque de datos, los archivos PDF o XML generados por cada factura que se genera. La tabla consta de 16 campos, de los cuales los más importantes son.

Tabla 28. Campos de la tabla fcpdfXXX

Campo	Tipo	Null	Key	Default	Descripción
PDOC	char(20)	SI	UNI		Nombre del documento
PENVMAIL	char(1)	SI	MUL	N	Enviado por mail (S/N)
PREIMPR	char(1)	SI	MUL	N	Reimpreso (S/N)
PFILEPDF	mediumblob	SI		NULL	Bloque de datos con el archivo
XUSUARIO	char(20)	SI			Usuario que crea el archivo
XFECALTA	date	SI		01/01/1980	Fecha de creación del archivo
PFECDOC	date	SI	MUL	01/01/1980	Fecha de la factura
PFECENVIO	date	SI		01/01/1980	Fecha de envió
PEMAIL	char(100)	SI			Email de envió
PCODCLI	int(11)	SI	MUL	0	Código de cliente
PNUMDOC	double(14,0)	SI	MUL	0	Número de factura
PPDFFIRM	char(1)	SI	MUL	N	Firmado digitalmente (S/N)
PTIPODOC	char(1)	SI	MUL	P	Tipo de documento (P-PDF/X-XML)

Tabla de proveedores fcproXXX

En esta tabla se almacenan los proveedores con los que la empresa trabaja. Entre otros datos se guardan los datos fiscales y bancarios de cada uno de los proveedores. La tabla tiene un total de 112 campos, de los cuales los más importantes y usados son los siguientes.

Tabla 29. Campos de la tabla fcproXXX

Campo	Tipo	Null	Key	Default	Descripción
PCOD	int(11)	SI	UNI	0	Código de proveedor
PNOM	char(50)	SI	MUL		Nombre del proveedor
PDOM	char(50)	SI			Dirección del proveedor
PCODPO	char(8)	SI	MUL		Código postal del proveedor
PPOB	char(40)	SI	MUL		Población del proveedor
PPAIS	char(20)	SI			Provincia del proveedor
PEXTRANJ	char(1)	SI		N	Extranjero (S/N)
PCEE	char(1)	SI		S	De la Comunidad Económica Europea (S/N)
PTEL1	char(20)	SI			Teléfono principal
PFAX1	char(20)	SI			Fax principal
PMAIL1	char(40)	SI			Email principal
PDNI	char(20)	SI	MUL		DNI o CIF del proveedor
POBSE	mediumtext	NO		NULL	Observaciones
ACTUALIZAD	char(1)	SI			Actualizado (S/N)
FECACTU	date	SI		01/01/1980	Fecha de actualización
XUSUARIO	char(20)	SI			Usuario que lo da de alta
XFECALTA	date	SI		01/01/1980	Fecha de alta
PFORPA	int(11)	SI		1	Forma de pago
PNOMCOMERC	char(50)	SI			Nombre comercial del proveedor
PBANCO	char(30)	SI			Banco de pago
PENT	char(4)	SI			Entidad
PSUC	char(4)	SI			Sucursal
PDIG	char(2)	SI			Digito de Control
PCUE	char(10)	SI			Número de cuenta
PRICONCED	double	SI		0	Riesgo concedido por el proveedor

Tabla de series de documentos fcserxxx

En esta tabla se almacenan por cada línea cada una de las series de documentos que se pueden crear en cada empresa. La tabla consta de 62 campos, de los cuales, los más importantes son.

Tabla 30. Campos de la tabla fcserXXX

Campo	Tipo	Null	Key	Default	Descripción
SCOD	int(11)	SI	UNI	0	Código de serie
SDES	char(30)	SI	MUL		Descripción
SPRIDOC	int(11)	SI		1	Primer documento
SULTDOC	int(11)	SI		899999	Último documento
SOBS	mediumtext	NO		\N	Observaciones
XUSUARIO	char(20)	SI			Usuario que lo crea
XFECALTA	date	SI		01/01/1980	Fecha de creación
SULTMODIF	timestamp	NO		\N	Fecha última actualización
SRECTIFI	char(1)	SI		N	Serie de facturas rectificativas (S/N)
SNOFACALB	char(1)	SI		N	Serie no facturable (S/N)
SINVSUJPAS	char(1)	SI		N	Serie de Inversión de sujeto pasivo (S/N)
SMEZSUJPAS	char(1)	SI		N	Permite mezclar Inversión de sujeto pasivo (S/N)

Tabla de stock de artículos por almacén fcstkxxx

En esta tabla se almacenan por cada línea el stock y la ubicación de cada artículo en cada almacén. La tabla consta de un total de 40 campos, de los cuales, los más importantes son.

Tabla 31. Campos de la tabla fcstkXXX

Campo	Tipo	Null	Key	Default	Descripción
ACODAR	char(28)	SI	MUL		Código de articulo
AALM	int(11)	SI	MUL	0	Almacén
ASTOCK	double	SI	MUL	0	Stock en el almacén
ASTAND	char(20)	SI			Ubicación en el almacén
AULTMODIF	timestamp	NO		NULL	Fecha última modificación

Tabla de usuarios fcusu

Esta es una tabla general que no depende de la empresa en la que se ejecute el programa, es común para todas. El contenido de cada registro representa todos los usuarios que tienen permiso para usar Xgestevo y cuáles son sus privilegios de acceso. La tabla consta de 128 campos, de los cuales los más importantes son.

Tabla 32. Campos de la tabla fcusu

Campo	Tipo	Null	Key	Default	Descripción
XNOMBRE	char(20)	SI	MUL		Nombre de usuario
XCONTRA	char(20)	SI			Contraseña de usuario
XACCTODO	char(1)	SI		S	Acceso total (S/N)
XEMPREINI	char(3)	SI			Código de empresa de inicio
XCAMEMP	char(1)	SI		S	Puede cambiar de empresa (S/N)
XSERIEINI	int(11)	SI		0	Serie de inicio
XCAMSER	char(1)	SI		S	Puede cambiar de serie (S/N)
XALMAINI	int(11)	SI		0	Almacén de inicio
XCAMALM	char(1)	SI		S	Puede cambiar de almacén (S/N)
XUSUARIO	char(20)	SI			Usuario que o da de alta
XFECALTA	date	SI		01/01/1980	Fecha de alta
XNOMCOMPL	char(50)	SI			Nombre completo de usuario
XULTMODIF	timestamp	NO		NULL	Fecha de última modificación
XCODIGOUNI	int(11)	NO	PRI	NULL	Id único autoincremental

Tabla de agencias de transporte fcvehXXX

En esta tabla se almacenan las agencias de transporte que tiene creadas la empresa. Cada una de las líneas contiene una agencia y, sobre todo, la ruta y el nombre del fichero que genera con los datos de expedición para que el software de la agencia en cuestión lo importe e integre en su sistema. Contiene además los códigos de bultos y palés asignados por cada agencia a la empresa para poder codificar cada envío. La tabla consta de 51 campos, de los cuales los más importantes son.

Tabla 33. Campos de la tabla fcvehXXX

Campo	Tipo	Null	Key	Default	Descripción
VCOD	int(11)	SI	UNI	0	Código de agencia
VDES	char(40)	SI	MUL		Nombre de la agencia
XUSUARIO	char(20)	SI			Usuario que la da de alta
XFECALTA	date	SI		01/01/1980	Fecha de alta
VCODBULTOS	char(20)	SI			Código para bultos
VCODPALET	char(20)	SI			Código para palé
VRUTAFIC	char(100)	SI			Ruta del fichero de expedición
VFIC	char(40)	SI			Nombre del fichero de expedición
VULTMODIF	timestamp	NO		NULL	Fecha de última modificación

Anexo 2. El almacén

Se ha dado un repaso a la base tecnológica sobre la que se va a sustentar este proyecto, pero no hay que olvidar que el objetivo de este módulo es el de facilitar las gestiones principales de un almacén.

A lo largo del resto del proyecto se va a hablar de muchos términos que para mucha gente pueden resultar desconocidos, o, aunque las palabras suenen, pueden resultar raras si se sacan del contexto de trabajos de un almacén. Hace falta un cierto conocimiento de base para entender muchas de las acciones que este software realiza, por lo que en las siguientes secciones se explicarán algunas de ellas.

Además, se verá cómo Xgestevo entiende y gestiona el almacén y cómo realiza las tres funciones principales que en todo almacén se realizan.

Finalmente, se verán qué soluciones actuales hay para el control del almacén en toda empresa que trabaje con Xgestevo, sus pros, sus contras y por qué se justifica la creación de este nuevo módulo de control de almacén.

Pero antes de pasar a los apartados descritos anteriormente, hay que plantearse dos preguntas. ¿Qué es un almacén? ¿Por qué es tan importante un férreo control de las operaciones en un almacén?

Para conocer lo que es un almacén vamos a utilizar la definición que hace Anaya Tejero, J.J. (2011) "*Etimológicamente, la palabra almacén sugiere una instalación específica para el albergue de productos de diferente naturaleza (materiales, productos comerciales, herramientas o utillaje en general, mobiliario, etc.); en definitiva, sería algo similar a lo que en la lengua anglosajona se conoce como warehouse*"

Tras esta definición general de lo que es un almacén, hay que especificar que, el tipo de almacenes para los que va destinado este software es a los que tienen como objetivo, el almacenaje, para posterior distribución de mercancía, con lo que se profundiza más en los procesos de entrada y salida del material. Aunque cualquier tipo de almacén podrá usarlo, aunque sea solo para el control del stock, pues todos los módulos serán independientes.

La segunda pregunta se puede comprender planteando una situación bastante real y que sería: un almacén de 6.000 metros cuadrados, en el que trabajen más de 50 personas, tenemos más de 30.000 referencias de artículos distintas repartidas por todo el almacén y un stock que ronda los 10 millones de euros. Queda bastante claro que, con este planteamiento, es muy importante saber dónde está ubicado el material en cada momento y

cuanto stock hay de cada artículo. No se puede usar la memoria, ni ningún otro soporte que no sea informático para almacenar ese conocimiento de manera eficiente.

Ahora hay que añadirle las operaciones rutinarias, en las cuales entra al día aproximadamente medio millón de euros más de material por las mañanas, que mandan los proveedores y sale por las tardes otro medio millón de material, de media, para los clientes. O lo que es lo mismo, el stock del almacén rota hasta un 5% todos los días (cosa no excesivamente rara en algunos sectores, como puede ser la informática).

Es difícil el pensar cómo llevar el día a día de un almacén de este tipo o de cualquier otro tamaño, más grande o incluso mucho más pequeño, sin tener muy bien controladas todas las operaciones que se realizan. De ahí surge la necesidad del software de gestión de almacén y su sustentación en una robusta base de datos que organice y permita el acceso rápido a la consulta, inserción y modificación de toda esta información que de otra manera sería imposible de controlar de una forma eficiente.

Los resultados de una mala gestión son obvios: malas ubicaciones de productos, que hace que no se encuentren lo suficientemente rápidos o que directamente no se encuentren y provoque una perdida para la empresa; mala o lenta gestión del material que llega del proveedor, que provoca que el producto no se pueda poner a la venta cuando posiblemente más necesario sea tener un stock visible del producto; y finalmente el que más repercusión puede tener para el cliente final, la mala gestión de las salidas, que puede provocar que al cliente no le llegue todo el material, que no llegue en buen estado o que directamente se mande a otra persona.

En las siguientes secciones se describe una a una estas operaciones que realiza el almacén y un poco después, como Xgestevo las gestiona de manera eficiente.

3.1 Operaciones básicas en un almacén

Dentro de un almacén existen gran cantidad de operaciones que se realizan día a día, relacionadas con el material, con la logística, incidencias, transportes, clientes, proveedores y muchas más. Pero, de entre todas, hay tres más importantes y sin las cuales un almacén no puede funcionar.

Estas operaciones son: el control sobre el stock, incluyendo el mantenimiento de los datos del artículo; las entradas de material al almacén, la relación y documentación que se realiza con los proveedores; y las salidas de material del almacén, operaciones con los clientes, los pedios y los albaranes.

3.1.1 Productos y el control de stock

Productos. También denominados Artículos o Ítems de un almacén. Se denomina producto a todo material que se encuentre en el almacén a la espera de su procesamiento, ya sea para la producción de otros productos más complejos o para la venta y envío a clientes.

Stock. También denominado Inventario o Existencias. Es el conjunto de todos los productos que hay dentro de un almacén. Cuando se habla del stock de un determinado producto, se refiere a la cantidad de este producto que se encuentra ubicado y disponible en el almacén.

El mantenimiento de los productos es una tarea fundamental y casi podría decirse que la más importante de cara al resto de procesos de la empresa, ya que, gracias a una buena gestión, se pueden tener características básicas de los productos como peso y dimensiones o incluso una imagen del producto, si aún no se tiene. Todas ellas muy útiles a la hora de la venta y de optimización de la logística y del transporte.

El tener permanentemente controlada la ubicación de cada producto y optimizada según categorías de rotación, o lo que lo mismo, la cantidad de veces que un producto es buscado por el personal del almacén, permite una optimización del tiempo que se emplea en la búsqueda de artículos, pues se tiende a colocar aquello que más se mueve, en las ubicaciones más cercanas a los puestos de trabajo del personal que prepara los paquetes.

Finalmente, el mantener un inventario continuo o saber en todo momento el stock que hay de un producto, permite planificar las compras y asegurar que cuando se va a ir a buscar un producto que se ha vendido o se necesita para manufacturar otro más complejo, se encontrará la cantidad que realmente se necesita y se esperaba encontrar.

3.1.2 Proveedores y las entradas de material

Proveedor. Toda aquella persona física o jurídica (empresa) que proporciona o vende los productos que más tarde cuando lleguen al almacén, pasaran a formar parte del stock.

Entrada de material. El proceso de entrada de material es aquel por el cual se realiza la agregación de los productos al stock del almacén. En un proceso simplificado de control de almacén, sería la única manera de sumar stock al inventario.

El proceso de entrada de material empieza mucho antes de la propia entrada, y consiste en la compra de los productos a un proveedor que desemboca en la realización de un pedido de compras.

Este pedido de compras resume todo el material que se ha pedido al proveedor, y es el sistema que utilizara el personal del almacén para verificar que el material que llega del proveedor es lo que se había pedido y que por tanto es correcto.

El acto de verificación del material finalizará con la creación de un documento llamado entrada de material y que no es más que un listado de todo el material que ha llegado y que por tanto puede entrar en el almacén

3.1.3 Clientes y las salidas de material

Cliente. Toda aquella persona física o jurídica (empresa) que adquiere el material del que se dispone en el almacén, normalmente como resultado de un proceso de compra.

Salida de material. El proceso de salida de material es aquel por el cual se realiza la sustracción de los productos al stock del almacén. En un proceso simplificado de control del almacén, sería la única manera de restar stock al inventario.

El proceso de salida de material también empieza con un proceso de compra, pero en este caso son los clientes quienes compran el material del que se dispone en el almacén. Este proceso de compra da lugar a un pedido de ventas.

Este pedido de ventas resume todo el material que el cliente ha comprado, es un listado de todo lo que hay que preparar para que el cliente se lo lleve o le sea enviado. Por norma general, el material en un pedido se encuentra en estado de reservado, por lo que, aunque aún pertenezca al stock, no será visible para el resto de departamentos, incluidos el de compras y el de ventas. Es el personal del almacén el que tendrá que buscar estos productos y prepararlos para su entrega.

Tras la preparación de todo el material del que consta el pedido, se realizará un documento llamado albarán (o salida) en el cual se resume todo el material que sale del almacén. Tras la creación de este albarán, el material desaparece definitivamente del stock del almacén.

Existe una modalidad especial de envío de material que últimamente se está extendiendo mucho, sobre todo en el mundo de la distribución. Este método de envío se denomina dropshipping y consiste en el envío del material a un tercero que queda fuera de la relación comercial entre el cliente y el vendedor. Este envío se realiza en nombre del cliente en lugar de aparecer el vendedor como remitente. Se usa mucho en la distribución de productos de informática, en las cuales, una tienda le compra un producto a un mayorista, y en lugar de recibir la tienda el producto y luego dárselo a su cliente, la tienda pide que le sea enviado el producto directamente al cliente. Para estos tipos de envío, el mayorista pone como datos de remitente todos los datos de la tienda, en lugar de sus datos, pues el cliente final no debe saber que el material viene de un sitio distinto que no sea la tienda a la que le ha comprado el material.

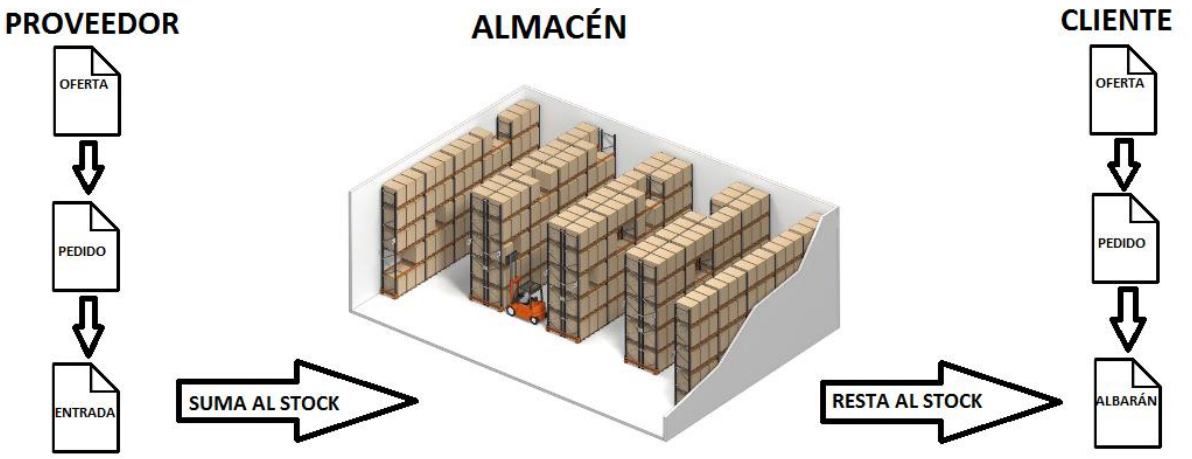

Ilustración 18. Operaciones del almacén