Grado en Ingeniería Informática

Trabajo Final de Grado

Pasarela de unión SEGA-DynamicsAX

Autor:
Albert Mestre Vicente

Supervisor:
Carlos Tena Monfort
Tutor académico:
Lledó Museros Cabedo

Fecha de lectura: 17 de Julio de 2019
Curso académico 2018/2019

Resumen

Pasarela de unión entre el ERP de Dynamics AX y el sistema gestor de almacenes SEGA. La pasarela de unión consiste en un sistema capaz de gestionar diferentes tareas de manera concurrente y paralela. Sobre este sistema se programan tareas que se encargan de coger los mensajes de comunicación procedentes de SEGA y transmitir esa información al sistema de AX a través de un servicio web. Del mismo modo, también se programan tareas para obtener información de la base de datos de AX y generar los ficheros de comunicación para SEGA. Todas estas tareas se realizan con una completa gestión de los errores. Dentro de este proyecto también se incluye el desarrollo del servicio web encargado de recibir los mensajes.

Palabras clave

Dynamics AX, Sistema gestor de almacenes, Gestión de tareas, ERP

Keywords

Dynamics AX, Warehouse management system, Task management, ERP

Índice general

Capítulo 1

Introducción

1.1. Contexto y motivación del proyecto

Marie Claire es una empresa que lleva más de 100 años dedicándose a la industria textil. Por lo que se refiere al departamento informático de la empresa, y mas concretamente a la sección de desarrollo, actualmente se enfrenta a un proyecto muy grande de cambio en el sistema de ERP, dentro del cual se incluye el proyecto que se va a describir a continuación. El proyecto consiste en desarrollar un sistema que sirva de pasarela de unión entre el Sistema Estándar de Gestión de Almacenes (SEGA) y el ERP de Microsoft Dynamics AX. Dentro de esta pasarela se incluye un servicio web del lado de AX para recibir la comunicación.

El sistema de SEGA se encarga de optimizar la utilización del espacio físico del almacén e informar de forma rápida acerca de las cantidades y la localización de los productos almacenados. Por otra parte Dynamics AX se encarga de gestionar la información y automatizar muchas de las prácticas de negocio asociadas con los aspectos operativos o productivos de la empresa.

Durante muchos años en la empresa se utilizó un ERP llamado Movex para la gestión de la información (inventario, ventas, producción, etc). Ya desde un inicio, el sistema de Movex se implementó con varias deficiencias estructurales y de rendimiento que provocan muchos problemas de funcionamiento y de pérdidas de la información, por tanto se ha decidido migrar toda la funcionalidad al nuevo sistema ERP de Dynamics AX.

En el momento que se empieza a desarrollar el proyecto de este documento, ya se han migrado los sistemas de producción y finanzas, y el sistema de compras está casi completado. De entre las funcionalidades que falta por migrar, se incluye la de gestión del inventario. Para poder migrar el sistema de inventarios es fundamental tener una pasarela de comunicación entre el sistema de control del almacén (SEGA) y Dynamics AX. Por tanto, la finalidad de este proyecto es poder migrar el control del inventario a AX.

Para gestionar el inventario de la empresa es necesario conocer el estado de los productos en el almacén, ahí es cuando se necesita la comunicación del sistema AX con SEGA, y para realizar esta comunicación es necesario replicar, en parte, el sistema de comunicación actual entre SEGA

y Movex. Estos dos sistemas tienen un protocolo de comunicación basado en ficheros de texto plano que se muestra en el Anexo A, donde se explica la codificación de la información de los cambios en el almacén. El cambio de sistema tiene que ser transparente desde el punto de vista de SEGA, es por eso que se tiene que mantener el sistema de comunicación que se usa actualmente y construir la pasarela en función de ello.

Por otra parte, el sistema de Dynamics AX recibirá las comunicaciones a través de un servicio web, el cual hay que crear e implementar en él los servicios necesarios para que pueda recibir los mensajes que afectan al inventario de la empresa y confirmar si se han podido realizar las acciones correctamente.

1.2. Objetivos y alcance del proyecto

En este apartado se van a comentar los objetivos principales que se pretenden lograr con el desarrollo del proyecto explicado en este documento. Además también se va a tratar el alcance del proyecto en sus diferentes ámbitos.

El objetivo principal del proyecto es realizar la pasarela de comunicación que atrape los mensajes provenientes de SEGA, los analice y los convierta en la información que necesite Dynamics AX para así mantener los dos sistemas comunicados y consistentes. Del mismo modo, también tiene que entender los mensajes de AX y convertirlos al protocolo de comunicación que entiende SEGA. Toda esta comunicación se tiene que realizar en un entorno a prueba de fallos donde los posibles errores queden registrados y se notifique al encargado de manera inmediata, por tanto será muy importante cuidar estos temas. Además, aunque la pasarela tenga actualmente un objetivo muy concreto, se prevé que en un futuro se le añada funcionalidad para otros ámbitos de la empresa, por tanto, el programa se tiene que desarrollar de manera que sea fácil añadir estos cambios en un futuro.

Por lo que refiere al alcance del proyecto, se cree bien definido desde el inicio, y se puede dividir en las secciones que se detallan a continuación.

1.2.1. Alcance Funcional

El sistema que se va a desarrollar ha de incluir la siguiente funcionalidad para cumplir con los requisitos deseados:

- El sistema ha de permitir leer los mensajes y entenderlos siguiendo la base del protocolo de comunicación que se tiene implantado actualmente entre SEGA y Movex. Existen distintos tipos de mensajes y cada uno de ellos se tienen que tratar de forma diferente y concurrentemente para lograr la mayor eficiencia posible.
- Junto con el sistema base habrá un servicio web en el lado de Dynamics AX que será capaz de recibir la información que el sistema base extraiga de los mensajes, para luego pasar esa información al ERP que se encargará de hacer las acciones pertinentes con dicha

información. Además el servicio web también tiene que contestar al sistema base con el resultado de su ejecución y cualquier tipo de error que se haya podido generar.

- Todo el sistema tiene que ser robusto y a prueba de fallos, por tanto, para evitar que se pierda información en caso de que ocurra algo en mitad de la comunicación o durante la ejecución de alguna orden en Dynamics AX, se tiene que crear un sistema de persistencia que permita saber con seguridad qué parte de los mensajes se ha tratado y cuál no. Además de eso, todos los errores producidos se tienen que documentar en algún sistema de *log* que permita mantener un registro y notificar al responsable del sistema que se ha producido un error.
- La pasarela debe estar funcionando el mayor tiempo posible para no entorpecer el desarrollo de la actividad de la empresa, por tanto se tiene que crear un sistema de *watchdog* para controlar las caídas del sistema y volver a encenderlo en caso de que eso ocurra.
- Todo el sistema de comunicación se tiene que poder controlar desde una interfaz gráfica que permita visualizar el estado de los mensajes que se han tratado o que se estén tratando en ese momento. Además se tiene que poder encender o apagar ciertos servicios en función de las necesidades. Junto con esto, la interfaz también tiene que permitir modificar los parámetros de ejecución de las tareas, tales como los directorios desde donde obtiene los mensajes, o la frecuencia en que se ejecutan cada uno de los servicios.
- En un futuro próximo se prevé incrementar la funcionalidad de este sistema, por tanto es muy importante mantener la modularidad y la abstracción del sistema, para que sea lo más sencillo posible añadir nuevos tipos de mensajes o nuevos servicios de comunicación.

1.2.2. Alcance Organizativo

La gestión del inventario es un apartado muy importante en una empresa que se dedica a la producción y venta de artículos, y está muy presente en muchos ámbitos de la empresa. Sin embargo, como este sistema solo gestiona la comunicación entre el gestor del almacén y el ERP de la empresa, su alcance organizativo está limitado. Por tanto, el sistema afecta principalmente al departamento de inventario de la empresa, aunque sí que se le puede atribuir un poco de implicación en los departamentos de producción y transporte ya que están muy relacionados con la gestión del inventario, y por ende, con el sistema de pasarela.

1.2.3. Alcance Informático

El sistema se tiene que conectar principalmente con otros dos sistemas para poder desarrollar su funcionalidad:

- Por una parte, el sistema tendrá que comunicarse con el sistema de SEGA, que se encarga de gestionar el almacén. Desde el punto de vista de la pasarela, SEGA es un sistema cerrado con el cual solo se puede interactuar a través de sus mensajes de comunicación. Tanto a los mensajes salientes como entrantes a SEGA, se accede a través de unos directorios concretos a los que si que tiene acceso libre la pasarela de unión.

- Por otra parte, el sistema tendrá que comunicarse con el ERP de Dynamics AX que gestiona la información de la empresa. Esta comunicación sí que es más libre ya que se tiene que desarrollar juntamente con la pasarela, por tanto se puede adaptar a las necesidades del momento. Principalmente la comunicación se realizará a través de uno o varios servicios web que estarán funcionando dentro del sistema de AX. Sin embargo, la base de datos que utiliza AX también es accesible para la pasarela de unión, por tanto, si la información que se necesita es simple, se puede leer directamente de la base de datos sin necesidad de que AX la gestione.

Capítulo 2

Planificación del proyecto

2.1. Metodología

En este apartado se van a explicar los diferentes motivos que se tuvieron en cuenta en el momento de decidir que metodología de desarrollo utilizar, además de argumentar las ventajas que proporciona la metodología utilizada frente a otras posibilidades.

La metodología de trabajo que se va a seguir durante el desarrollo está basada en el desarrollo en cascada. Se han tenido en cuenta diferentes factores a la hora de decantarse por este tipo de desarrollo. Primero, el equipo de desarrollo está formado por una sola persona, por tanto no es necesario realizar reuniones planificadas regularmente para controlar el avance de diferentes miembros de un equipo, más allá de las reuniones de control que realicé con el supervisor.

También los requisitos del sistema están muy bien definidos y no se prevén cambios durante su desarrollo, por tanto, es sencillo planificar su desarrollo con antelación y diseñar el sistema sin que se tenga que modificar su diseño durante el desarrollo; y por último, el sistema no se va a poder utilizar hasta que no esté toda su funcionalidad implementada y probada, por tanto, no tiene sentido centrarse en desarrollar incrementos utilizables como consiguen las metodologías ágiles. Parece más sencillo desarrollar y realizar una sola puesta en marcha cuando esté completamente terminado.

Ahora bien, para la parte del desarrollo del sistema, se van a utilizar herramientas propias de la metodología ágil para compensar algunas de las carencias de la metodología en cascada y hacer el desarrollo del sistema mas cómodo. En concreto se va a utilizar un *tablero kanban* para agilizar la distribución del trabajo y lograr un desarrollo más fluido.

Por otra parte, como la empresa no se dedica al desarrollo de productos software, el departamento de desarrollo informático es pequeño. Esto implica que no existe ninguna metodología de desarrollo muy compleja. En ese sentido, la empresa proporciona libertad para decidir cómo se quiere planificar el desarrollo. El único punto destacable que se puede aplicar a este proyecto sobre la metodología de desarrollo en la empresa, son las reuniones que se realizan aproximadamente cada dos semanas y en las que participan los informáticos de la empresa y algunos

trabajadores que puedan estar interesados en algún producto informático. En estas reuniones se discuten temas sobre si algún producto nuevo es necesario o no, o el resultado que esté logrando otro producto, pero nunca se ha hablado sobre cómo se tiene que desarrollar el producto.

Por tanto, la elección de la metodología de desarrollo que se sigue en este proyecto fue completamente decisión propia, dentro de las posibilidades que la empresa permite. Siguiendo los criterios expuestos anteriormente, se decidió por utilizar una metodología en cascada, tal como se mostrará en el apartado siguiente, aunque utilizando herramientas propias de las metodología ágiles por comodidad y utilidad.

2.2. Planificación

En este apartado se va a explicar como se ha adaptado la planificación en cascada a este proyecto de desarrollo de software. Esto se hará, primeramente, explicando que características propias del desarrollo en cascada se han utilizado en la planificación. Seguidamente se presentará la planificación elegida y el porque de su estructura.

Para el proyecto de este documento, se pensó en utilizar la planificación en cascada de manera bastante fiel, ya que, como se ha comentado en el apartado anterior, las características del proyecto parecían idóneas para ello. Por tanto, la planificación tendrá una fase de Definición de requisitos, una fase de análisis, otra fase de diseño y finalmente, una fase de desarrollo y pruebas.

Al ser un proyecto desarrollado en una estancia de prácticas en empresa, a estas fases habrá que añadir, una sección de aprendizaje, donde se aprenden los conceptos necesarios para el desarrollo del producto en la empresa, y que se realizará durante el transcurso de la estancia.

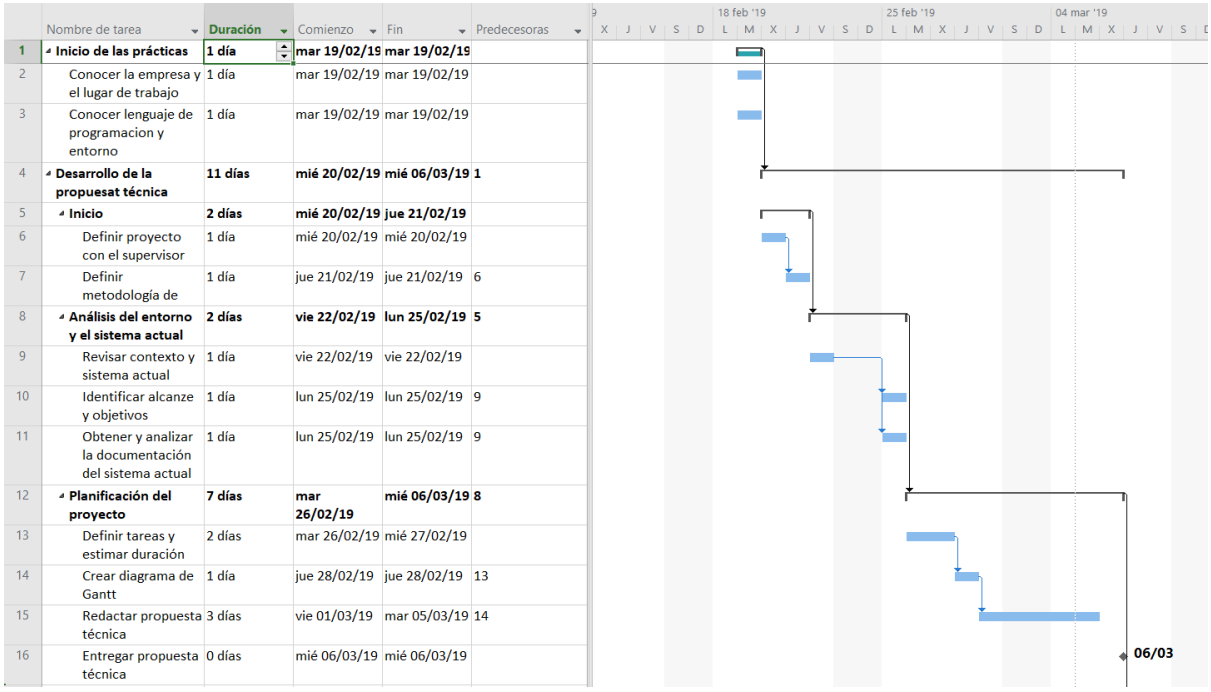
El desglose de las tareas del proyecto, junto a su planificación temporal y sus dependencias se puede ver en la Figura 2.1 (Inicio y propuesta técnica), en la Figura 2.2 (Requisitos, análisis y diseño) y en la Figura 2.3 (Desarrollo y puesta en marcha).

2.2.1. Inicio y propuesta técnica

Como podemos ver en la Figura 2.1, los primeros días de la estancia se dedicaron a conocer la empresa y su entorno de trabajo informático. Al ser esta una empresa conocida por el alumno y utilizar un lenguaje de programación (C#) muy similar al utilizado durante el grado de Ingeniería Informática, no fue necesario dedicar mucho tiempo. Se consideró suficiente con dedicar la jornada de un día de 4 horas.

Con esto también se tomó la decisión de que, en caso de necesitar realizar aprendizaje de alguna tecnología nueva para el alumno, se realizaría a lo largo del desarrollo de la estancia. Esto fue así porque no se preveían necesarias muchas lecciones y no tenía sentido dedicar muchas horas inicialmente al no haber muchas tecnologías que necesitaran aprendizaje.

Tras el apartado anterior, se planificó el desarrollo de la propuesta técnica. Esta propuesta



	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
1	**Inicio de las prácticas**	**1 día**	**mar 19/02/19**	**mar 19/02/19**	
2	Conocer la empresa y el lugar de trabajo	1 día	mar 19/02/19	mar 19/02/19	
3	Conocer lenguaje de programacion y entorno	1 día	mar 19/02/19	mar 19/02/19	
4	**Desarrollo de la propuesat técnica**	**11 días**	**mié 20/02/19**	**mié 06/03/19**	**1**
5	**Inicio**	**2 días**	**mié 20/02/19**	**jue 21/02/19**	
6	Definir proyecto con el supervisor	1 día	mié 20/02/19	mié 20/02/19	
7	Definir metodología de	1 día	jue 21/02/19	jue 21/02/19	6
8	**Análisis del entorno y el sistema actual**	**2 días**	**vie 22/02/19**	**lun 25/02/19**	**5**
9	Revisar contexto y sistema actual	1 día	vie 22/02/19	vie 22/02/19	
10	Identificar alcanze y objetivos	1 día	lun 25/02/19	lun 25/02/19	9
11	Obtener y analizar la documentación del sistema actual	1 día	lun 25/02/19	lun 25/02/19	9
12	**Planificación del proyecto**	**7 días**	**mar 26/02/19**	**mié 06/03/19**	**8**
13	Definir tareas y estimar duración	2 días	mar 26/02/19	mié 27/02/19	
14	Crear diagrama de Gantt	1 día	jue 28/02/19	jue 28/02/19	13
15	Redactar propuesta técnica	3 días	vie 01/03/19	mar 05/03/19	14
16	Entregar propuesta técnica	0 días	mié 06/03/19	mié 06/03/19	

Figura 2.1: Diagrama de Gantt donde se muestra la planificación del inicio de las prácticas y el desarrollo de la propuesta técnica. La duración se muestra en días con jornadas de 4 horas al día

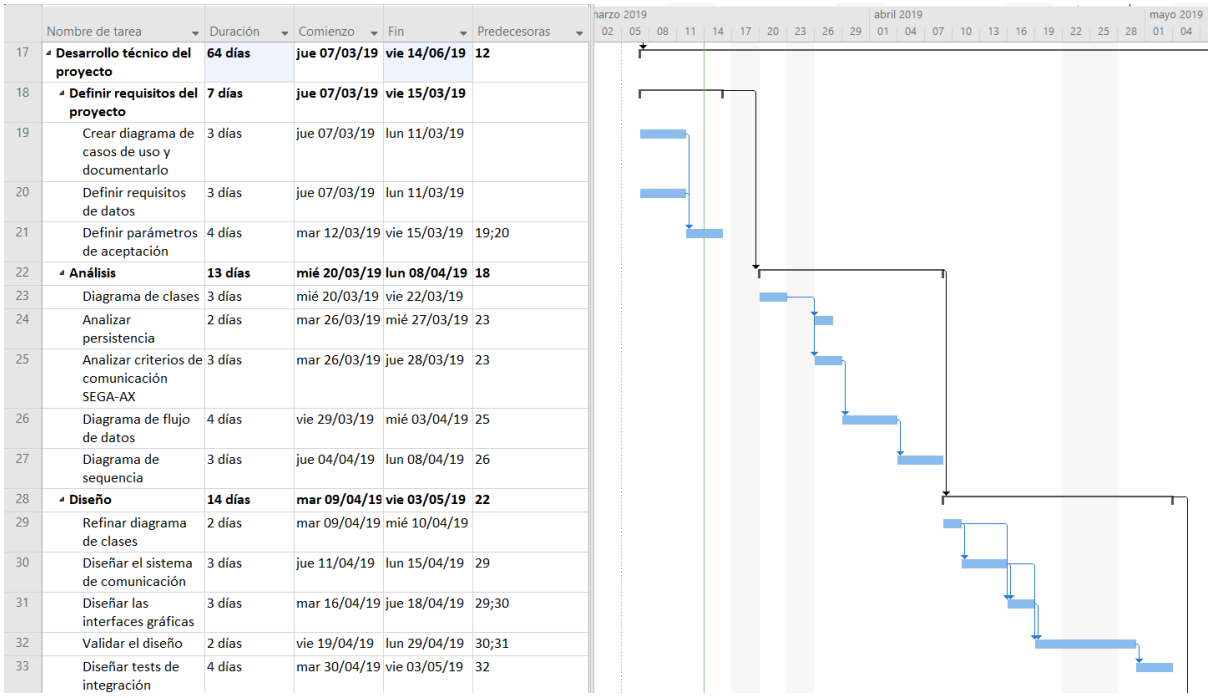
serviría para comprender completamente los objetivos del proyecto y su necesidad dentro de la empresa. Este apartado se consideró muy importante ya que sería la base de conocimiento a partir de la cual se empezaría a analizar y diseñar el proyecto, por tanto, se consideraron necesarios 11 días de 4 horas cada uno para su desarrollo.

Para lograr el desarrollo de la propuesta técnica, primeramente se dedicarían dos días a poner de acuerdo el proyecto y la metodología de trabajo entre el alumno y el supervisor. Después de eso, se dedicarían otros dos días para que el alumno pudiese comprender los sistemas que serían afectados por el nuevo sistema a desarrollar, lo cual permite identificar mejor el contexto y el alcance del proyecto.

Finalmente, se emplearían siete días para la planificación del proyecto como tal. Primeramente se estimaron las tareas que se debían realizar y su duración. Esto permitió realizar el diagrama de Gantt que se muestra en las figuras nombrada en este apartado, además de servir de base para redactar la propuesta técnica.

2.2.2. Definición de requisitos, análisis y diseño

Con la propuesta técnica ya entregada, se pasaría al desarrollo técnico del proyecto. Para este desarrollo se estimó una duración de 64 días, ya que sería el grueso del proyecto y en lo que se dedicaría más tiempo. (Los detalles de cada una de las fases del desarrollo se mostrarán en los siguientes apartados de la memoria).



	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
17	**Desarrollo técnico del proyecto**	**64 días**	**jue 07/03/19**	**vie 14/06/19**	**12**
18	**Definir requisitos del proyecto**	**7 días**	**jue 07/03/19**	**vie 15/03/19**	
19	Crear diagrama de casos de uso y documentarlo	3 días	jue 07/03/19	lun 11/03/19	
20	Definir requisitos de datos	3 días	jue 07/03/19	lun 11/03/19	
21	Definir parámetros de aceptación	4 días	mar 12/03/19	vie 15/03/19	19;20
22	**Análisis**	**13 días**	**mié 20/03/19**	**lun 08/04/19**	**18**
23	Diagrama de clases	3 días	mié 20/03/19	vie 22/03/19	
24	Analizar persistencia	2 días	mar 26/03/19	mié 27/03/19	23
25	Analizar criterios de comunicación SEGA-AX	3 días	mar 26/03/19	jue 28/03/19	23
26	Diagrama de flujo de datos	4 días	vie 29/03/19	mié 03/04/19	25
27	Diagrama de sequencia	3 días	jue 04/04/19	lun 08/04/19	26
28	**Diseño**	**14 días**	**mar 09/04/19**	**vie 03/05/19**	**22**
29	Refinar diagrama de clases	2 días	mar 09/04/19	mié 10/04/19	
30	Diseñar el sistema de comunicación	3 días	jue 11/04/19	lun 15/04/19	29
31	Diseñar las interfaces gráficas	3 días	mar 16/04/19	jue 18/04/19	29;30
32	Validar el diseño	2 días	vie 19/04/19	lun 29/04/19	30;31
33	Diseñar tests de integración	4 días	mar 30/04/19	vie 03/05/19	32

Figura 2.2: Diagrama de Gantt donde se muestra la planificación para el análisis y diseño del sistema. La duración se muestra en días con jornadas de 4 horas al día

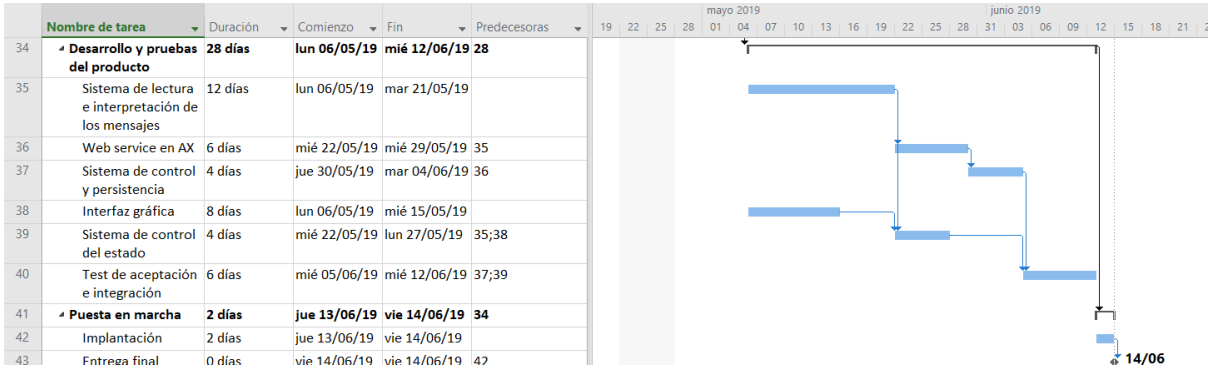
El primer paso sería definir los requisitos del proyecto de manera más técnica, mediante la creación y documentación de los diagramas pertinentes. A esta tarea se dedicarían siete días. Se consideró suficiente con ese tiempo ya que mucha de la información necesaria ya se había obtenido durante las fases anteriores.

Con los requisitos ya definidos se pasaría al análisis del sistema, donde se estudiarían los componentes necesarios para el desarrollo del sistema. Esta fase se consideró con más peso que la anterior, ya que requería de un mayor conocimiento del entorno, y por tanto se estimó su duración en 13 días. Durante esta fase se realizarían los diagramas propios de un análisis de software, lo cual implicaría conocer mejor el entorno del sistema y permitiría concretar muchas de las necesidades del sistema.

Una vez el análisis del sistema esté completado, se comenzaría con el diseño del sistema. Aunque el diseño suele ser una tarea más costosa que el análisis, como muchas de las decisiones importantes ya se habrían tomado durante la fase anterior, se consideró que con 14 días seria suficiente para terminar. Durante esta fase se documentaría la arquitectura del sistema y sus interacciones con los demás sistemas afectados, además de documentar su resultado. Durante esta fase también se diseñarían los test de integración que se utilizarían mas adelante en el desarrollo.

2.2.3. Desarrollo del sistema y pruebas

El último paso para el desarrollo técnico del proyecto, sería realizar el desarrollo del sistema y de las pruebas necesarias para asegurar su funcionamiento. Esta fase se consideró la mas costosa



	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
34	**Desarrollo y pruebas del producto**	**28 días**	**lun 06/05/19**	**mié 12/06/19**	**28**
35	Sistema de lectura e interpretación de los mensajes	12 días	lun 06/05/19	mar 21/05/19	
36	Web service en AX	6 días	mié 22/05/19	mié 29/05/19	35
37	Sistema de control y persistencia	4 días	jue 30/05/19	mar 04/06/19	36
38	Interfaz gráfica	8 días	lun 06/05/19	mié 15/05/19	
39	Sistema de control del estado	4 días	mié 22/05/19	lun 27/05/19	35;38
40	Test de aceptación e integración	6 días	mié 05/06/19	mié 12/06/19	37;39
41	**Puesta en marcha**	**2 días**	**jue 13/06/19**	**vie 14/06/19**	**34**
42	Implantación	2 días	jue 13/06/19	vie 14/06/19	
43	Entrega final	0 días	vie 14/06/19	vie 14/06/19	42

Figura 2.3: Diagrama de Gantt donde se muestra la planificación para el desarrollo técnico y la puesta en marcha del sistema. La duración se muestra en días con jornadas de 4 horas al día

e importante del proyecto y por tanto se le estimó una duración de 28 días. Cabe destacar que, como se verá en un apartado posterior, esta es la fase que mas se desvió de su plan inicial durante el desarrollo. Otro factor que se tuvo en cuenta para dar una duración mayor a esta fase fue que se preveía que se tendría que dedicar algún tiempo para aprender algunas de las posibles tecnologías nuevas para el alumno necesarias para el desarrollo, como puede ser la programación en un ERP.

Para elegir el orden de implementación de las diferentes partes del sistema, se planteó desarrollar primero la base del programa y dejar las comunicaciones con los otros sistemas para cuando el sistema base ya estuviera terminado. La interfaz gráfica del programa se planteó desarrollarse en paralelo al resto del sistema ya que al ser una interfaz simple, no requeriría de mucho esfuerzo. Más detalles de la implementación se comentarán posteriormente en el apartado de Implementación y pruebas.

2.3. Estimación de recursos y costes del proyecto

La estimación de los recursos necesarios y los costes que pueda tener un proyecto es un apartado muy importante en la planificación del desarrollo de software. Una mala estimación de los costes puede llevar a que un proyecto no se considere viable aunque sí lo sea, o por el contrario, que se estime menos costoso de lo real y no se pueda llegar a terminar por falta de presupuesto. Por tanto, en este apartado se va a realizar un estudio sobre los costes del proyecto según el modelo basado en puntos de casos de uso. Primeramente se comentará el porqué de la elección de este modelo frente a posible alternativas, luego se realizará la estimación de los costes siguiendo los criterios del modelo elegido, y finalmente se añadirá el coste de los recursos necesarios para el desarrollo y los costes indirectos de la actividad a realizar.

Para la elección del modelo a seguir, se plantearon dos opciones, por un lado el modelo elegido basado en puntos de casos de uso, y por otro lado el modelo de COCOMO. Ambos modelos tienes sus ventajas y desventajas, sin embargo, el problema con el modelo de COCOMO para este proyecto es que esta basado en proyecto históricos y en el número de líneas de código que suelen tener proyectos similares. En el caso de este proyecto, al no ser una empresa que se dedique explícitamente al desarrollo de software, no se dispone de suficientes datos históricos

Tipo de actor	Descripción	Peso
Simple	Otro sistena externo, interactua con el sistema a desarrollar mediante una interfaz de programación definida y conocida	1
Medio	Otro sistema externo que interactúa a través de protocolo de comunicación	2
Complejo	Un usuario físico que interactúa a través de una interfaz gráfica de usuario	3

Cuadro 2.1: Parámetros para calificar el peso de los casos de uso según sus interacciones

como referencia. En cambio, el modelo basado en puntos de casos de uso no requiere de datos históricos, sino que basa sus cálculos en los casos de uso del sistema y sus actores, información de la que sí se dispone.

Por tanto, para realizar la estimación del coste siguiendo el modelo basado en puntos de casos de uso [1], primero se obtiene el cálculo de los puntos de casos de uso no ajustados (UUCP, Unajusted Use Case Points). Consiste en calcular el peso tanto para actores (UAW, Unajusted Actor Weights) como para casos de uso (UUCW, Unajusted Use Case Weights) y sumar el resultado de UAW y UUCW. Los criterios para la asignación de pesos de los actores se muestran en la Tabla 2.1. Los actores que se han detectado para este proyecto de desarrollo de software se muestran a continuación (Una mayor explicación de los actores se puede encontrar en el apartado de Diseño y Análisis):

- **Administrador** Este actor será un usuario experto en la informática e interactuará con el sistema a través de una interfaz gráfica, que aunque no tiene que ser muy completa, sí lo suficiente para que se pueda usar cómodamente y no se olvide su funcionamiento tras varias semanas sin usarse. Por tanto se considera de **Coste complejo**
- **SEGA** Este actor es un sistema externo que se comunica con el sistema a desarrollar a través de un protocolo de comunicación basado en ficheros de texto. Como es un protocolo de comunicación muy concreto y no del todo intuitivo, se considera de **Coste medio**
- **Dynamics AX** Este actor es un sistema externo que se comunica con el sistema a desarrollar a través de un servicio web, lo cual es una interfaz de programación conocida y definida. Por tanto, se considera de **Coste bajo**

Con estos valores ya podemos calcular el UAW nombrado anteriormente siguiendo la siguiente ecuación:

$UAW = \sum_{i=1}^{(n-actores)} (cantidadTipoActor_{\mathrm{i}} * peso)$

Con esto obtenemos un UAW de 6 puntos.

El siguiente paso sera obtener los pesos de los casos de uso de este proyecto. Los criterios para la asignación de pesos de los casos de uso se muestran en la tabla 2.2. Los casos de uso que se han detectado para este proyecto de desarrollo de software se muestran a continuación (Una mayor explicación de los casos de uso se puede encontrar en el apartado de Diseño y Análisis):

Tipo de caso de uso	Descripción	Peso
Simple	El caso de uso contiene 3 o menos transacciones	5
Medio	El caso de uso contiene entre 4 y 7 transacciones	10
Complejo	El caso de uso contiene más de 7 transacciones	15

Cuadro 2.2: Parámetros para calificar el peso de los casos de uso según su número de transacciones

1. El sistema ha de transmitir el maestro de artículos que tiene la base de datos de AX y que necesita SEGA. Esto implicará acceder a la base de datos de AX, volcar la información en un fichero de texto siguiendo el protocolo adecuado y dejar ese fichero en el directorio de SEGA. Un total de 3 transacciones que dan un caso de uso **Simple**.

2. El sistema ha de transmitir regularmente la información de los últimos movimientos de inventario que se han generado en SEGA que afectan al inventario de AX. Esto implicará acceder a los directorios de SEGA para obtener el mensaje y descomponer la información de dentro de ese mensaje según el protocolo de comunicación, mandar esa información a través del servicio web de AX. Dentro de AX hay que obtener esa información para generar una cabecera del diario de transferencias y las diferentes líneas. Un total de 3 transacciones que dan un caso de uso **Simple**.

3. El sistema ha de transmitir diariamente por la noche el stock total que tiene el almacén de SEGA en ese momento. Esto implicará acceder a los directorios de SEGA para obtener el mensaje y descomponer la información de dentro de ese mensaje según el protocolo de comunicación, luego mandar las diferencias de stock a AX a través del servicio web para generar una cabecera del diario de transferencias y las diferentes líneas. Un total de 3 transacciones que dan un caso de uso **Simple**.

4. El sistema ha de notificar al gestor del almacén cuando se realiza un movimiento desde el almacén de Villafranca al almacén de Borriol y confirmar que el envío haya llegado correctamente según la respuesta del gestor del almacén. Esto implica acceder a la base de datos de Dinamics AX para comprobar los productos en tránsito y con destino el almacén de Borriol, con esa información hay que generar un fichero de texto donde se vuelca la información y dejarlo en el directorio de SEGA. Posteriormente, cuando el envío llegue al almacén, hay que acceder al directorio de SEGA para obtener el fichero de confirmación, descomponer la información de dentro del fichero y mandar a AX a través del servicio web. Un total de 4 transacciones que dan un caso de uso **Medio**.

5. El sistema ha de notificar al almacén cuando se ha de realizar un envío de productos. Esto también implica que el sistema de AX debe recibir la confirmación de la preparación del envío para poder realizar el albarán de envío y así poder expedir las mercancías. Esto implica acceder a la base de datos de AX para ver los envíos pendientes, con esta información hay que realizar un fichero de comunicación para depositarlo en el directorio de SEGA. Posteriormente, cuando se haya terminado de preparar el envío, hay que acceder otra vez al directorio de SEGA para obtener el fichero con la confirmación de la preparación, y mandar esa información a AX para que prepare el albarán de envió. Un total de 5 transacciones que dan un caso de uso **Medio**.

6. El sistema debe implementar un control de su propia ejecución para que se reinicie el sistema en caso de cuelgue o bloqueo. Esto implica acceder a la lista de procesos del

programa para comprobar si se está ejecutando, acceder a un directorio para comprobar si se ha generado el fichero de control y acceder al directorio con los ejecutables del sistema. Un total de 3 transacciones que dan un caso de uso **Simple**.

7. El sistema debe notificar al encargado cualquier error crítico que se produzca durante su ejecución, además de registrar la incidencia en algún *log*. Este caso de uso se ejecuta para todos los demás. Esto implica acceder al sistema de correos de la empresa y al directorio donde se encuentras los ficheros de log. Un total de 2 transacciones que dan un caso de uso **Simple**.

8. El sistema debe permitir al encargado gestionar la ejecución de los servicios, desde encenderlos y apagarlos hasta cambiar su configuración. Esto implica acceder al fichero de configuración para cambiar los parámetros. Un total de 1 transacción que da un caso de uso **Simple**.

Según los criterios expuestos anteriormente, se obtiene un total de 6 casos de uso simples y 2 casos de uso medios. Lo cual, aplicando la siguiente formula, significa un total de 50 puntos.

$$UUCW = \sum(tipoCasoDeUso_{\mathrm{i}} * peso)$$

Por tanto, para obtener el UUCP, solamente tenemos que sumar los valores obtenidos según la siguiente formula:

$$UUCP = UAW + UUCW$$

Lo cual nos da como resultado total un UUCP de 56 puntos.

El siguiente paso es calcular los puntos de casos de uso (UCP, Use Case Points). Consiste en realizar el producto de los puntos de casos de uso no ajustados, el peso de los factores técnicos (TCF, Technical Factors) mostrados en la Tabla 2.3 y el peso de los factores ambientales (EF, Enviroment Factors) mostrados en la Tabla 2.4, los cuales se ponderan respecto a las habilidades y experiencias del grupo de trabajo. Para calcular los puntos, se deben considerar valores entre 0 y 5, donde: Irrelevante de 0 a 2, Medio de 3 a 4, Esencial 5.

Para el desarrollo del sistema de este documento, se han considerado los siguientes puntos de factores técnicos:

- **T1**: El sistema solo va a estar funcionando simultáneamente en un mismo servidor, por tanto no necesita nada relacionado con un sistema distribuido. Obtiene una puntuación de **0 puntos**.

- **T2**: Al no ser una aplicación que se esté usando continuamente por usuarios no es necesaria una respuesta inmediata, sin embargo, el programa tiene que ser eficiente en la ejecución de sus servicios. Obtiene una puntuación de **2 puntos**.

- **T3**: El sistema no va a ser usado mucho por usuarios finales, solo en contadas ocasiones, eso sí, en el momento de ser usado debería permitir trabajar cómodamente. Obtiene una puntuación de **2 puntos**.

Factor	Descripción	Peso
T1	Sistema distribuido	2
T2	Objetivos de rendimiento o tiempo de respuesta	2
T3	Eficiencia del usuario final	1
T4	Procesamiento interno complejo	1
T5	El código debe ser reutilizable	1
T6	Facilidad de instalación	0.5
T7	Facilidad de uso	0.5
T8	Portabilidad	2
T9	Facilidad de cambio	1
T10	Concurrencia	1
T11	Objetivos especiales de seguridad	1
T12	Acceso directo a terceras partes	1
T13	Es necesaria formación específica	1

Cuadro 2.3: Parámetros para calificar el peso de los factores técnicos

- **T4**: El sistema no realiza cálculos complejos en su funcionalidad, la carga de trabajo recae más en las comunicaciones y en la multitarea. Obtiene una puntuación de **1 punto**.

- **T5**: Uno de los objetivos del proyecto es desarrollar un sistema que sea ampliable y reutilizable, por tanto, este criterio es importante, aunque tampoco es completamente esencial si algún componente resulta muy específico. Obtiene una puntuación de **4 puntos**.

- **T6**: La instalación implicará dejar de utilizar otros sistemas, por tanto, requerirá de cierto trabajo de adecuación. Esto puede complicar en parte la instalación del nuevo sistema. Obtiene una puntuación de **3 puntos**.

- **T7**: El sistema solo va a ser utilizado por personal informático, por tanto no es necesaria mucha facilidad de uso. Sin embargo, como seguramente se use en intervalos de tiempo separados, es necesario que se pueda recordar su funcionamiento tras un tiempo sin usarse. Obtiene una puntuación de **2 puntos**.

- **T8**: El sistema esta diseñado para ser usado en un servidor con el sistema operativo Windows y no se tiene previsto modificar este criterio, por tanto, la portabilidad no es muy necesaria. Obtiene una puntuación de **0 puntos**.

- **T9**: Uno de los criterios del sistema era que se pudiera ampliar fácilmente su funcionalidad en un futuro. Aunque la nueva funcionalidad no debería ser muy diferente de la inicial, si que es un criterio muy importante su facilidad de cambio. Obtiene una puntuación de **4 puntos**

- **T10**: El sistema trabaja con servicios que funcionan de manera concurrente, aunque, como ninguno de ellos se debería comunicar entre sí ni compartir memoria local, la concurrencia se facilita bastante. Obtiene una puntuación de **3 puntos**.

- **T11**: El sistema no requiere de ningún tipo de seguridad ya que funcionará sobre un servidor que no tiene acceso de terceros. Se supone que la seguridad ya viene dada por el sistema del servidor. Obtiene una puntuación de **0 puntos**.

Factor	Descripción	Peso
E1	Familiaridad con el modelo de proyecto	1.5
E2	Experiencia en la aplicación	0.5
E3	Experiencia en orientación a objetos	1
E4	Capacidades de análisis	0.5
E5	Motivación personal	1
E6	Estabilidad en los requerimientos	2
E7	Personal de medio tiempo	-1
E8	Dificultad del lenguaje de programación	-1

Cuadro 2.4: Parámetros para calificar el peso de los factores ambientales

- **T12**: El objetivo del sistema es prestar comunicación entre sistemas propios de la empresa, por tanto, no hay ninguna tercera parte que tenga acceso directo. Obtiene una puntuación de **0 puntos**.
- **T13**: El sistema está pensado para ser usado por usuarios informáticos especializados, y al no tener una interfaz muy complicada, será suficiente con una explicación básica de su funcionamiento. Obtiene una puntuación de **1 punto**

Con los valores asignados ya se puede obtener el TCF según la siguiente formula:

$TCF = 0{,}6 + (0{,}01 * \sum(peso * puntuacion))$

Esto implica que el sistema obtiene un peso de factores técnicos total de **0,815**.

El siguiente paso es calcular la puntuación que se otorga a los factores ambientales según el criterio del equipo de desarrollo. Para el desarrollo del sistema de este documento, se han considerado los siguientes puntos de factores ambientales:

- **E1**: Se tiene experiencia teórica con este tipo de modelo de proyecto, en cambio, la experiencia práctica no es del todo extensa. Obtiene una puntuación de **3 puntos**.
- **E2**: Se tiene experiencia en las aplicaciones de trabajo concurrente y en las que se basan en comunicación entre sistemas, sin embargo, el ámbito de trabajo del software empresarial es más desconocido. Obtiene una puntuación de **3 puntos**.
- **E3**: Los lenguajes orientados a objetos son principalmente en los que más experiencia se tiene. Obtiene una puntuación de **5 puntos**.
- **E4**: La experiencia teórica sobre el análisis se considera extensa, sin embargo, en el ámbito práctico se considera que todavía queda margen de mejora. Obtiene una puntuación de **3 puntos**.
- **E5**: Aunque no es un tipo de proyecto muy llamativo, al formar parte de un trabajo final de grado y el primer proyecto real para una empresa, hay un nivel de motivación elevado. Obtiene una puntuación de **4 puntos**.
- **E6**: Los requisitos parecen estables inicialmente, aunque no es del todo seguro que no se pueda llegar a modificar alguno de ellos. Obtiene una puntuación de **4 puntos**.

- **E7**: Salvo en algunas ocasiones limitadas, el proyecto se desarrolla completamente en jornadas de medio tiempo. Obtiene una puntuación de **4 puntos**.

- **E8**: El lenguaje que se utiliza para el desarrollo es completamente familiar y no genera ningún problema ni dificultad. Obtiene una puntuación de **0 puntos**.

Con los valores asignados ya se puede obtener el EF según la siguiente formula:

$EF = 1{,}4 + (-0{,}03 * \sum(peso * puntuacion))$

Esto implica que el sistema obtiene un peso de factores ambientales total de **0,785**.

Con estos valores ya se puede calcular el tamaño del proyecto en casos de uso (UCP) de acuerdo a la siguiente formula:

$UCP = UUCP * TCF * EF = 56 * 0,815 * 0,785 = 35,83$

Este valor de 35,83 indica el tamaño del proyecto en puntos de casos de uso, para obtener el coste en tiempo y monetario del proyecto, necesitamos calcular el esfuerzo en Horas-Persona, de acuerdo a la siguiente formular.

$Horas - Persona = UCP * Factordeproductividad$

Para el factor de productividad, se supone un valor de 20, tal y como recomienda el autor del modelo Karner [1]. Esto genera un esfuerzo en horas-persona de **716 horas**. Estas son las horas de trabajo necesarias para el desarrollo, que equivalen a **18 semanas** de trabajo, y como se supone que se desarrollaría por un solo programador, hay que multiplicar las horas necesarias por el sueldo medio de un programador junior, que se estima en 1400€al mes. Esto provoca un coste en recursos humanos de **6270€**, que añadiendo impuestos del 20 % resultaría en un total de **7524€**.

A estos costes hay que añadir los gastos indirectos que se generan por el simple hecho de estar trabajando, como podrían ser la luz o la tarifa de internet. Estos gastos indirectos se suelen estimar un 20 % del coste total del proyecto, por tanto, el presupuesto del proyecto ascendería a **9.028€**.

Si fuera necesario, a este presupuesto habría que añadir el coste proporcional del hardware necesario para su desarrollo, en el caso de este proyecto, no se requiere ya que todo lo necesario ya esta en la empresa. Tanto el ordenador que se utiliza para el desarrollo como el servidor donde funcionará el sistema ya se encontraba en la empresa, por tanto no es necesario ningún prorrateo. Por lo que se refiere al software de desarrollo, se han utilizado software de uso libre como el Visual Studio.

Hay que destacar que la estimación realizada da al proyecto una duración en horas-persona de 716 horas, sin embargo, el proyecto se planificó para desarrollarse en 300 horas de estancia en prácticas. Aunque puede parecer incongruente, hay que pensar que de esas 716 horas, parte del trabajo ya lo realizó la empresa antes de iniciar la estancia en prácticas, como por ejemplo, el analizar el problema y plantear una solución, la cual luego se desarrollaría por el estudiante

ID	Descripción del Riesgo	Tipo de Riesgo
R01	Requisitos poco completos	Riesgo del producto
R02	Baja temporal del único desarrollador del proyecto	Riesgo del proyecto
R03	Baja temporal del único supervisor del proyecto	Riesgo del proyecto
R04	Falta de experiencia en el funcionamiento del software ERP de una empresa	Riesgo del producto
R05	Entorno inestable en el ámbito informático de la empresa	Riesgo del proyecto
R06	Diseño erróneo	Riesgo del proyecto
R07	Situación económica de la empresa	Riesgo del proyecto
R08	Cambio del lugar de trabajo semanalmente	Riesgo del proyecto

Cuadro 2.5: Riesgos encontrados para este proyecto y su tipo

en la estancia. También hay que comentar que, como se ve en el apartado de seguimiento del proyecto, no se han podido completar las fases de desarrollo de los test de integración, y la puesta en marcha. Completar estas fases que faltan aumentaría la duración real del proyecto, lo cual haría que se acercara más a las estimaciones realizadas.

2.4. Plan de riesgos

Todo proyecto de desarrollo de software es susceptible de sufrir contratiempos en su planificación inicial que dificulten seguir el plan trazado, y este proyecto en concreto no es una excepción. En este apartado de listarán los diferentes riesgos que se apreciaron en el proyecto, además de proponer medidas preventivas y paliativas para que los posibles riesgos tengan el menor impacto posible. La lista con los principales riesgos detectados se pueden ver en la tabla 2.5.

2.4.1. Análisis de los riesgos

R01: Requisitos poco completos

Magnitud: Variable según la fase en la que se detecten los problemas, baja si se resuelven en el inicio o el análisis del diseño, alta si se detectan en la fase de desarrollo.

Descripción: Aunque los requisitos parecen estar muy claros inicialmente, no se puede estar seguro de que al realizar el análisis del sistema algunas de esos requisitos se consideren inviables o cambien parcialmente debido a nuevas consideraciones.

Impacto: La incorporación o modificación de requisitos durante el desarrollo requerirá realizar cambios sobre gran parte de la documentación del producto elaborada con anterioridad al

momento del cambio. Estas modificaciones serán menos costosas durante las dos primeras fases del proyecto, pero pueden suponer modificaciones importantes durante las fases de Implementación y pruebas, pues no sólo cambiaría la documentación sino también el código fuente y los ejecutables.

Indicadores: Durante algunas de las fases del proyecto surgen problemas logísticos que ni el alumno ni el supervisor son capaces de resolver sin considerar de cambiar los requisitos del sistema.

R02: Baja temporal del único desarrollador del proyecto

Magnitud: Muy alta, ya que paraliza completamente el desarrollo del proyecto.

Descripción: En caso de que el alumno que está desarrollando el proyecto tuviera que dejarlo temporalmente por problemas de salud o cualquier otro motivo, el proyecto quedaría totalmente paralizado y la planificación perdería toda su utilidad.

Impacto: El proyecto no podría continuar su desarrollo y se tendrá que retrasar la planificación o alargar su duración para poder completar el proyecto

Indicadores: Ninguno, al ser un riesgo por causas externas al proceso, se supone que es un riesgo difícil de predecir.

R03: Baja temporal del único supervisor del proyecto

Magnitud: Alta, ya que aunque no paraliza directamente el proyecto, sí que lo relentizaría notablemente.

Descripción: Al ser el supervisor el único que comprende todo el conjunto del sistema del ERP en el que se apoya el sistema a desarrollar, habrá muchas decisiones de integración que no se podrán tomar por falta de información.

Impacto: Se tendrán que retrasar partes del desarrollo hasta que el supervisor vuelva a estar disponible o si no se tendrá que tomar decisiones que posiblemente sean erróneas y se tengan que cambiar posteriormente.

Indicadores: Ninguno, al ser un riesgo por causas externas al proceso, se supone que es un riesgo difícil de predecir.

R04: Falta de experiencia en el funcionamiento del software ERP de una empresa

Magnitud: Media-Baja, la falta de experiencia se puede solventar de varias maneras.

Descripción: Los programas ERP tienen su propia manera de funcionar muy enfocada a la forma de funcionar de una empresa, y aunque su programación utiliza un lenguaje muy similar al conocido por el alumno, hay varios conceptos de su funcionamiento, (como la forma que tiene de guardar la información en la base de datos o como funcionan sus servicios web) que si no se comprenden pueden entorpecer su desarrollo.

Impacto: La falta de experiencia provocaría un pequeño retraso en el desarrollo al tener que perder algún tiempo en formarse o apoyarse en alguna guía especializada para comprender mejor el funcionamiento.

Indicadores: El propio alumno se daría cuenta de su falta de experiencia al no comprender como funciona o tardar demasiado en realizar algunas de las tareas que requieren de programar en el ERP.

R05: Entorno inestable en el ámbito informático de la empresa

Magnitud: Alta, ya que un cambio en algún otro ámbito de la empresa, por ejemplo el departamento de producción, puede afectar al funcionamiento del almacén y por tanto, a la pasarela de unión.

Descripción: Aunque el sistema a desarrollar solo afecta al departamento de almacenamiento de la empresa, está relacionado con otros como el de producción o el de transporte. Un cambio en la forma que tienen de funcionar estos otros departamento, especialmente si también afecta a la programación del ERP, puede suponer un cambio brusco en la forma de funcionar de la pasarela. Estos cambios es posible que se den ya que actualmente todo el sistema informático de gestión de la empresa se esta trasladando al nuevo ERP.

Impacto: Los posibles cambios que pudieran surgir afectarían a todas las fases del desarrollo del proyecto, desde su análisis hasta su desarrollo, dependiendo del momento en el que se den.

Indicadores: En las reuniones que se realizan aproximadamente cada dos semanas y donde surgen las propuestas de cambios en el sistema informático de la empresa, se pueden observar los posibles cambios de los departamentos. Además, como esos cambios normalmente los realiza el supervisor del proyecto, este puede mantener informado al alumno.

R06: Diseño erróneo

Magnitud: Medio, un diseño erróneo puede retrasar el proyecto a causa de tener que repetir las fases de diseño y análisis, sin embargo, se puede modificar en cuanto se detecte el problema.

Descripción: Como el diseño se realiza en un etapa temprana del proyecto, es posible que llegado el momento el alumno todavía no tenga un conocimiento adecuado de como funciona el sistema completo, al ser el ámbito de una empresa grande desconocido para el. Esto podría implicar que el análisis se realizara con algunos errores que se tendrían que modificar e un futuro durante otras fases.

Impacto: Si se detectan errores se tendría que volver atrás en el desarrollo y modificar las fases anteriores para reflejar los cambios realizados.

Indicadores: Los fallos en el diseño se deberían detectar en fases posteriores, sobre todo durante la fase de diseño, al encontrar errores que se tiene que corregir para poder lograr el desarrollo del proyecto.

R07: Situación económica de la empresa

Magnitud: Muy baja, una mala situación económica de la empresa podría implicar una falta de recursos necesarios para el desarrollo, como podrían ser licencias de programas necesarios para desarrollar software.

Descripción: Aunque la empresa actualmente tiene una economía estable, en ocasiones si que ha tenido problemas financieros que han llevado a recortes y falta de recursos. Por tanto, aunque no es muy probable, esa situación se podría llegar a repetir.

Impacto: La falta de recursos económicos podría implicar tener que utilizar otros programas de uso libre, lo cual implicaría tener que aprender como funcionan. También podría generar recortes en el hardware utilizado, como es el servidor donde se supone que se va a implantar el sistema.

Indicadores: En las reuniones quincenales es fácil que se comenten esos recortes. Sino los propios compañeros de departamento deberían de comentar el problema si ocurre.

R08: Cambio del lugar de trabajo semanalmente

Magnitud: Baja, cambiar de lugar de trabajo puede suponer olvidos y un poco de baja de productividad, pero en una medida reducida.

Descripción: El desarrollo del proyecto se realiza durante 3 días a la semana en Borriol y 2 días en Villafranca. El cambio de ubicación puede suponer olvidos de información física importante y menos contacto con el supervisor. Sin embargo, como el desarrollo se realiza en un ordenador portátil y la empresa tiene un buen sistema de comunicaciones interno, este problema se reduce bastante.

Impacto: La falta de comunicación y los cambios del entorno de trabajo pueden provocar bajadas de rendimiento y tiempos muertos, pero de baja intensidad.

Indicadores: En los días de después del cambio de localización se es menos productivo que el día anterior, o se aprecian muchos tiempos muertos a causa del cambio de lugar de trabajo.

2.4.2. Acciones de prevención y corrección

R01: Requisitos poco completos

Plan de prevención: Realización de varias reuniones con el supervisor del proyecto donde se planteen la mayor cantidad de preguntas posibles sobre el futuro sistema para lograr la mayor comprensión posible sobre el. En caso de que algún requisito cambiara, intentar reflejar los cambios lo antes posible.

Plan de corrección: Intentar modificar los requisitos de manera que afecten lo menos posible al sistema ya planteado. En caso de que los cambios generen una modificación mas radical, realizar los cambios necesarios lo antes posible en el desarrollo.

R02: Baja temporal del único desarrollador del proyecto

Plan de prevención: Dentro de lo posible, intentar no hacer ninguna acción que comprometa la salud o que generen consecuencias que impidan continuar con el desarrollo del proyecto.

Plan de corrección: Como el desarrollo se realiza con un ordenador portátil, en caso de no poder acudir a la oficina para el desarrollo, sería posible realizarlo desde casa o el lugar donde se esté reposando.

R03: Baja temporal del único supervisor del proyecto

Plan de prevención: Como no se puede evitar este tipo de riesgo por su naturaleza, se intentaría planificar con antelación el trabajo a realizar para un par de semanas en adelante, así en caso de fallo, se podría seguir trabajando durante un tiempo.

Plan de corrección: Si se tiene planteado el trabajo para un par de semanas, no seria muy grave la baja del supervisor. Si no es el caso, se intentaría realizar otra parte del proyecto que no requiriera de la supervisión, como por ejemplo, el diseño de la interfaz gráfica, que es mas libre al criterio del alumno.

R04: Falta de experiencia en el funcionamiento del software ERP de una empresa

Plan de prevención: Durante la fase de definición de requisitos, el alumno debería ir interesándose en el funcionamiento de un software ERP para así encontrar cuanto antes las posibles dudas. Además también se podría ir obteniendo material de referencia para consultar cualquier posible duda puntual en caso de que el supervisor no este disponible.

Plan de corrección: En caso de detectar una falta de conocimientos importantes sobre el funcionamiento, se realizaría un tutorial o guía para intentar comprender mejor el funcionamiento. En caso de no ser posible esto, se realizaría el desarrollo con más ayuda por parte del supervisor.

R05: Entorno inestable en el ámbito informático de la empresa

Plan de prevención: Dentro de lo posible, intentar aislar lo más posible el diseño del sistema para que no se vea afectado por los cambios del entorno. Si esto no es posible, se intentaría estar lo más atento posible a los cambios para realizar las modificaciones lo antes posible incluso prever algunos posibles cambios y diseñar el sistema acorde a las posibles modificaciones.

Plan de corrección: En caso de necesitar realizar los cambios, se modificaría la documentación que ya se haya realizado para adaptarse al nuevo sistema.

R06: Diseño erróneo

Plan de prevención: Conseguir todo el conocimiento posible sobre él funcionamiento tanto del sistema futuro como de los sistemas a los que afecta para así lograr el mejor diseño posible.

Plan de corrección: En caso de diseño erróneo, se realizarían los cambios de manera que afectaran lo menos posible al sistema que ya este desarrollado, y se modificaría la documentación anterior en consecuencia.

R07: Situación económica de la empresa

Plan de prevención: Obtener desde un principio las licencias y material necesario para el desarrollo, para no depender así de más recursos en un futuro.

Plan de corrección: En caso de necesitar algún recurso que no se pudiera permitir por la empresa, se buscarían alternativas sin coste, o se intentaría adaptar a los recursos de los que se dispone actualmente.

R08: Cambio del lugar de trabajo semanalmente

Plan de prevención: Tener toda la información necesaria en la nube de la empresa para siempre tenerla disponible independientemente de donde se este trabajando. Revisar con la ayuda de una check list todo lo necesario que se tiene que coger para cambiar de lugar de trabajo.

Plan de corrección: En caso de algún olvido se puede pedir ayuda a los compañeros de departamento que estén trabajando en el otro lugar.

2.5. Seguimiento del proyecto

En esta sección se va a presentar el seguimiento del proyecto. También se van a comentar los cambios que se produjeron en la planificación y el porqué de esos cambios. Para mostrar estos cambios se van a mostrar el Gantt de seguimiento que se ve en la Figura 2.4 y la Figura 2.5. La fase del desarrollo de la propuesta técnica no se muestra ya que es anterior a la planificación del proyecto y, por tanto, no sufrió ninguna modificación. Por tanto, los cambios que se produjeron durante la fase del desarrollo técnico del proyecto son los siguientes:

- En la fase de **Definición de requisitos**, la duración total de la fase se mantuvo igual, sin embargo se cambiaron las dependencias de las tareas 19 y 20 para hacerlas más cortas pero desparalelizarlas. Se consideró que era mejor centrarse primero en una de ellas y luego en la otra. Con esto también se acorto un día la definición de los parámetros de aceptación, ya que esta tarea se completó más fácilmente.

- En la fase de **Análisis**, la duración total de esta fase se redujo en 4 días. La principal razón de ello es que se eliminó la realización del diagrama de flujo de datos, ya que este no se consideró necesario dado que los diagramas de secuencia ya mostraban mejor las acciones del sistema.

  Además, como el análisis de la comunicación entre SEGA y AX se terminó antes, se dió el día extra para realizar los diagramas de secuencia, por tanto, la duración de esta tarea aumentó en 1 día

- En la fase de **Diseño**, la duración total se redujo a 9 días. La principal razón de ello es que tanto la tarea de diseñar la interfaz gráfica como los tests de integración redujeron su duración.

La tarea de diseñar la interfaz gráfica se redujo a un solo día, dado que el sistema esta enfocado a un usuario experimentado y además, dispone de pocas pantallas. El esfuerzo para su diseño fue menor de lo esperado. En cuanto a los test de integración, la mitad de ellos aproximadamente se supusieron para probar la integración con SEGA, pero como SEGA no tiene un entorno de pruebas, la mitad del trabajo de esta tarea no se pudo realizar y por tanto su duración se redujo.

- En la fase de **Desarrollo y pruebas**, se amplió la duración a 36 días. Este aumento es posible dadas las horas que disminuyeron de las fases anteriores. Esta fase también ha sufrido cambios en el orden de las tareas. Durante la fase de diseño se decidió implementar primero la base del programa que gestionara las tareas, luego implementar las tareas partiendo de la funcionalidad programada en la tarea anterior, y finalmente programar el servicio web.

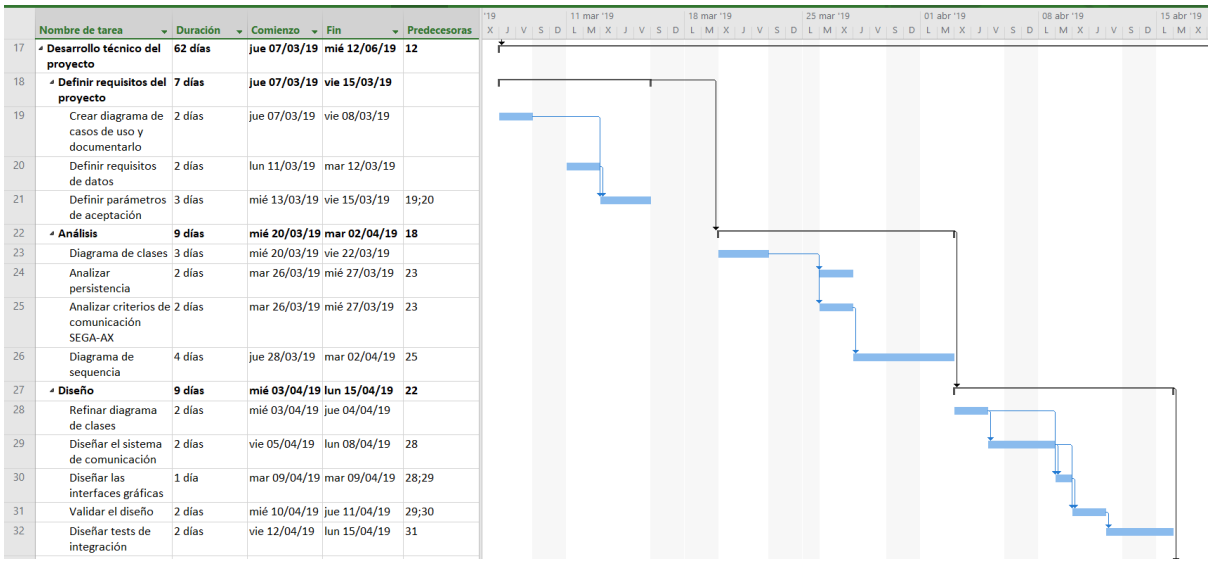
  Se sustituyó la tarea de control del estado por el sistema de control de tareas, al cual se dedicaron 12 días. Parte de la funcionalidad del sistema de lectura también se incluye dentro de esta tarea, por tanto su duración se vio ampliada considerablemente.

  En el sistema de lectura de los mensajes de comunicación, como parte de la funcionalidad ya estaba en el programa base y solo faltaba implementar los algoritmos de lectura, su duración se redujo a 6 días. A la par con estas dos tareas se implementó la interfaz gráfica. Esta decisión se mantuvo desde el inicio pero se alargó su duración al alargarse las otras tareas.

  Por lo que refiere a la implementación del servicio web, su duración se amplió a 12 días y hay varios motivos para ello. Primero, porque su método de implementación no era tan conocido y por tanto necesitó de un poco de aprendizaje, además, al formar parte de un entorno más grande, había que tener muchas cosas en cuenta durante el desarrollo y era un tema más delicado. En paralelo al desarrollo del servicio web también se desarrolló el sistema de persistencia ya que este también estaría dentro del sistema de AX y, por tanto, tenía más sentido desarrollarlo a la par.

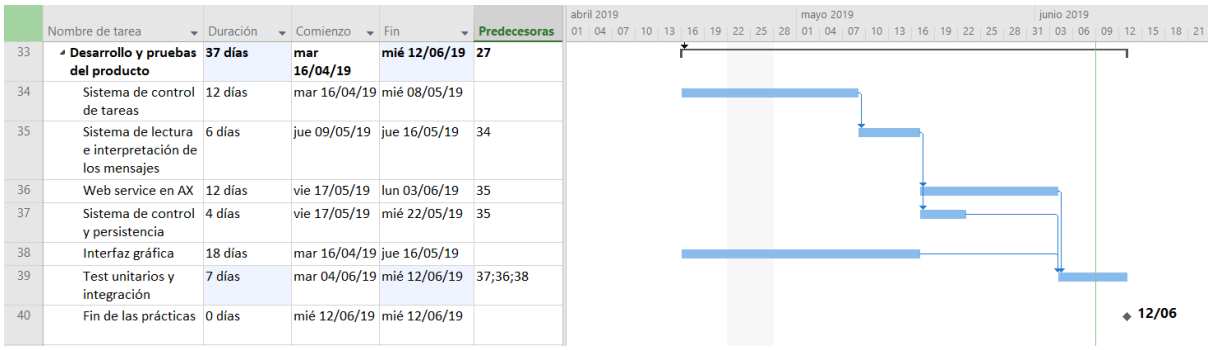
  Finalmente, los test se reconsideraron a test unitarios e integración. Su duración se amplió ligeramente al incluir los test unitarios, dando lugar a una duración de 7 días.

- En la fase de **Puesta en marcha**, esta fase no se pudo llegar a desarrollar dentro de la duración de la estancia de prácticas. El principal motivo es que el entorno de la empresa todavía no está preparado para dejar de utilizar el sistema antiguo de ERP para empezar a utilizar el de Dynamics AX. Se entiende que el resto del sistema está terminado, sin embargo, es muy posible que a la hora de la puesta en marcha en un futuro cercano, se tenga que modificar parte del sistema ya que la integración con SEGA no se ha podido probar a causa de no tener este un entorno de pruebas.



	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
17	**Desarrollo técnico del proyecto**	**62 días**	**jue 07/03/19**	**mié 12/06/19**	**12**
18	**Definir requisitos del proyecto**	**7 días**	**jue 07/03/19**	**vie 15/03/19**	
19	Crear diagrama de casos de uso y documentarlo	2 días	jue 07/03/19	vie 08/03/19	
20	Definir requisitos de datos	2 días	lun 11/03/19	mar 12/03/19	
21	Definir parámetros de aceptación	3 días	mié 13/03/19	vie 15/03/19	19;20
22	**Análisis**	**9 días**	**mié 20/03/19**	**mar 02/04/19**	**18**
23	Diagrama de clases	3 días	mié 20/03/19	vie 22/03/19	
24	Analizar persistencia	2 días	mar 26/03/19	mié 27/03/19	23
25	Analizar criterios de comunicación SEGA-AX	2 días	mar 26/03/19	mié 27/03/19	23
26	Diagrama de sequencia	4 días	jue 28/03/19	mar 02/04/19	25
27	**Diseño**	**9 días**	**mié 03/04/19**	**lun 15/04/19**	**22**
28	Refinar diagrama de clases	2 días	mié 03/04/19	jue 04/04/19	
29	Diseñar el sistema de comunicación	2 días	vie 05/04/19	lun 08/04/19	28
30	Diseñar las interfaces gráficas	1 día	mar 09/04/19	mar 09/04/19	28;29
31	Validar el diseño	2 días	mié 10/04/19	jue 11/04/19	29;30
32	Diseñar tests de integración	2 días	vie 12/04/19	lun 15/04/19	31

Figura 2.4: Diagrama de Gantt de seguimiento donde se muestra la planificación de análisis y diseño que se ha seguido realmente.La duración se muestra en días con jornadas de 4 horas al día



	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
33	**Desarrollo y pruebas del producto**	**37 días**	**mar 16/04/19**	**mié 12/06/19**	**27**
34	Sistema de control de tareas	12 días	mar 16/04/19	mié 08/05/19	
35	Sistema de lectura e interpretación de los mensajes	6 días	jue 09/05/19	jue 16/05/19	34
36	Web service en AX	12 días	vie 17/05/19	lun 03/06/19	35
37	Sistema de control y persistencia	4 días	vie 17/05/19	mié 22/05/19	35
38	Interfaz gráfica	18 días	mar 16/04/19	jue 16/05/19	
39	Test unitarios y integración	7 días	mar 04/06/19	mié 12/06/19	37;36;38
40	Fin de las prácticas	0 días	mié 12/06/19	mié 12/06/19	

Figura 2.5: Diagrama de Gantt de seguimiento donde se muestra la planificación de desarrollo y pruebas que se ha seguido realmente.La duración se muestra en días con jornadas de 4 horas al día

Capítulo 3

Análisis y diseño del sistema

En este capítulo se van a presentar los resultados de las fases de análisis y diseño del sistema, que darán forma a la estructura del sistema y que sirven de guía para el desarrollo y la implementación del sistema. Para eso, primeramente se van a definir los requisitos del sistema.

3.1. Definición de requisitos

- El sistema ha de permitir al encargado gestionar todos los aspectos relacionados con la ejecución de los servicios. Por una parte ha de poder cambiar en tiempo real y de forma simple las siguientes características de los servicios:
  - La frecuencia en la que se repite su ejecución o la hora del día en la que se tiene que ejecutar, en función del tipo de servicio que sea.
  - El texto que representa la descripción del servicio.
  - El directorio desde donde coger documentos, en caso de que el servicio lo requiera.
  - El directorio donde depositar documentos creados, en caso de que el servicio lo requiera.
  - El directorio raíz a partir del cual el servicio creará los documentos y directorios que necesite para su ejecución.
- El sistema ha de transmitir el maestro de artículos que tiene la base de datos de AX y que necesita SEGA. Este maestro de artículos contiene un listado con todos los artículos y sus variantes que se tienen registrados en la empresa. Esa información se encuentra actualizada en varias tablas de la base de datos de Dynamics AX y se tiene que replicar diariamente y durante la noche a SEGA. Para ello se requerirá un servicio que realize el trasvase de información siguiendo el protocolo de comunicación establecido por SEGA.
- El sistema ha de transmitir regularmente los últimos movimientos de inventario que se han realizado en el almacén de SEGA. El gestor de almacén tiene un listado con los últimos movimientos que se han registrado, donde se enumera qué productos se han movido y en qué cantidades. Para que tanto el gestor del almacén como el ERP estén con los datos

sincronizados, la información de los movimientos se tiene que replicar de SEGA a AX, leyendo los mensajes que SEGA genera y descodificando la información de acuerdo al protocolo de comunicación concreto para, posteriormente, pasar la información a AX a través de su servicio web. El desarrollo de la lógica del servicio web también se tiene que desarrollar y se incluiría dentro de este requisito.

- El sistema ha de transmitir diariamente por la noche el estado actual del almacén de SEGA, es decir, la lista de productos que tiene almacenados, sus cantidades y si alguno de ellos está en estado de preparación o distribución, para así tener los dos sistemas sincronizados. Este requisito puede parecer algo redundante con el requisito anterior, pero es necesario dada la forma de funcionar que tiene el gestor del almacén. El sistema deberá captar los mensajes que contienen esta información, comparar su contenido con la información actual que se tiene en el sistema de AX y igualar sus cantidades. Se debe suponer que en caso de diferencia, el gestor del almacén será el que tiene las cantidades correctas, y las cantidades de AX se deberán corregir.

- El gestor del almacén debe ser avisado cuando un envío de productos llegue al almacén. Más concretamente cuando se espere recibir un conjunto de productos o materiales procedentes del almacén de producción situado en Villafranca. El movimiento de productos entre almacenes lo realizan empleados de la empresa usando la funcionalidad que les proporciona AX. Cuando un envío se genera en el sistema del ERP, se debe informar al gestor de almacén con un fichero de comunicación que sigue el protocolo de comunicación que tiene SEGA. Este requisito también incluye comprobar la confirmación de recepción que SEGA genera cuando un envío esperado es recibido satisfactoriamente. Si el envío llega correctamente se tiene que notificar a AX para que se cierre la transacción. En caso contrario, se tiene que notificar del fallo o las cantidades perdidas a algún responsable.

- El gestor del almacén debe ser notificado cuando haya un envío pendiente de preparación, para que así la gente que trabaja en el almacén pueda preparar la comanda. Para ello, el sistema debe acceder regularmente a los envíos pendientes y si encuentra alguno, se lo debe notificar a SEGA para que este empiece la preparación. Cuando el envío ya este preparado para expedir, SEGA genera un fichero que se utiliza para avisar a Dynamics AX de que hay un envío a la espera de salir y necesita el albarán de envío y más información. Una vez AX ya ha generado esa información, el envío es expedido y se completa la operación.

- El sistema debe mantenerse en ejecución el mayor tiempo posible. Para lograr esto se tiene que implementar un sistema que controle la ejecución y el estado de la pasarela. En caso de que el sistema se cuelgue o se cierre de manera inesperada, se debe reiniciar manteniendo el estado anterior. Además, se debe notificar al encargado del error sucedido y las posibles causas. El error también debe quedar registrado en algún mecanismo de log que tenga el sistema para estudiar la posible causa con más detenimiento en el futuro.

- El sistema debe mantener la información de los posibles errores que se produzcan o advertencias que se generen en algún sistema de log. Además, si en algún momento algún servicio ha sufrido alguna incidencia, se tiene que apreciar a simple vista para que el supervisor lo vea fácilmente.

- Dado que la funcionalidad del sistema se plantea ampliar en un futuro cercano, el programa debe desarrollarse con esto en mente, es decir, en caso de querer añadir nuevas funcionalidades que en base sean similares a los servicios actuales, se debe de poder hacer con poco esfuerzo.

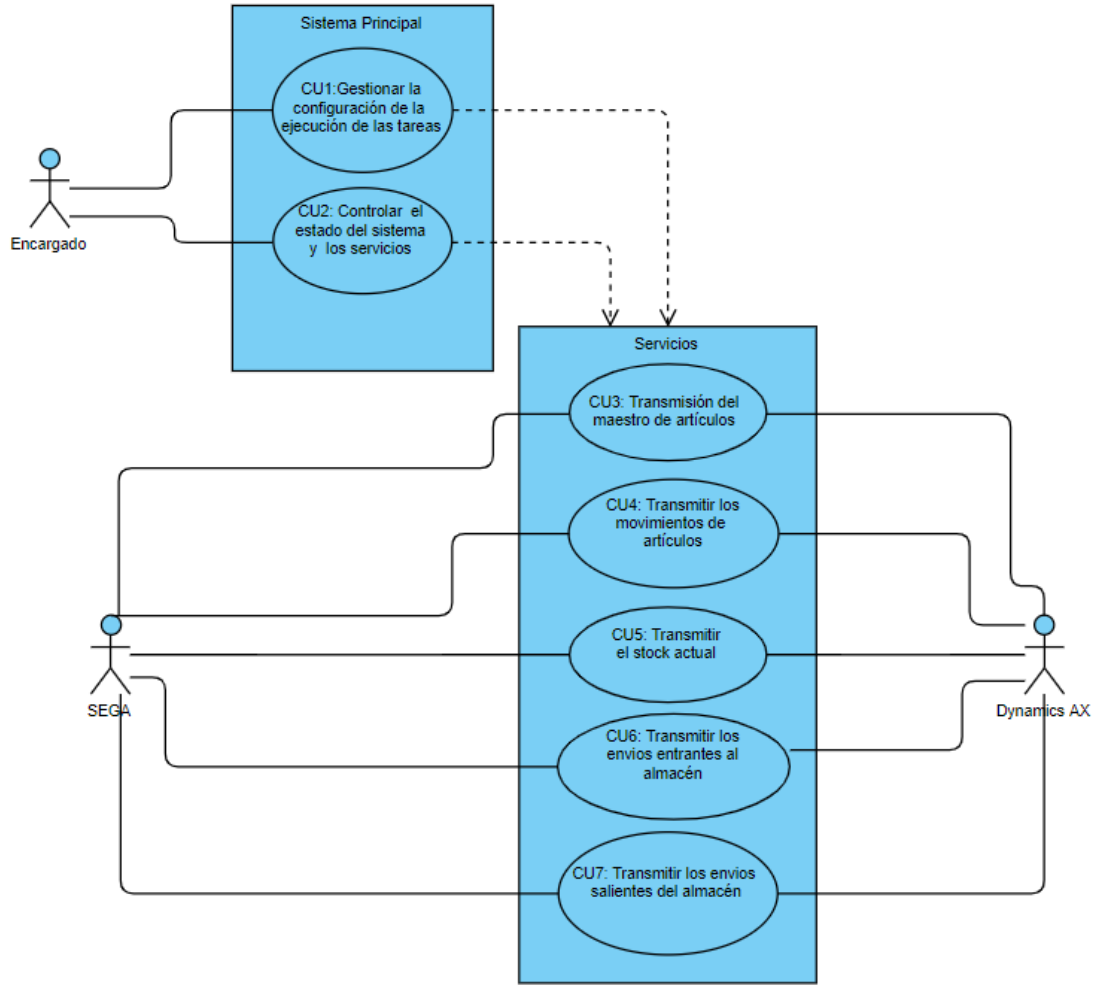


Figura 3.1: Diagrama de casos de uso del sistema

A partir de los requisitos anteriormente mencionados, se puede obtener el diagrama de casos de uso de la Figura 3.1, que nos muestra de manera simplificada la estructura del sistema final. A continuación se van a presentar las especificaciones de cada uno de los casos de uso del sistema:

1. **Identificador:** CU1.Gestionar la configuración de la ejecución de las tareas
   - **Descripción:** El encargado de controlar el sistema debe ser capaz de cambiar la configuración con la que se ejecutan los servicios. Cuando una configuración se cambia, el sistema debe adaptar la ejecución de los servicios acorde a la nueva configuración sin necesidad de reiniciar el sistema.
   - **Alcance:** Comienza en el momento que el encargado accede a la opción de cambiar la configuración de las tareas y termina en el momento que sale o guarda los cambios.
   - **Nivel:** Tarea pricipal
   - **Actor principal:** Encargado
   - **Actores secundarios:** Ninguno
   - **Relaciones:** CU3, CU4, CU5, CU6, CU7
   - **Precondición:** El sistema debe tener algún servicio configurado.
   - **Condición fin éxito:** El sistema modifica los servicios con la nueva configuración y la guarda.
   - **Condición fin fracaso:** El sistema no puede modificar la configuración porque no es correcta.
   - **Secuencia normal:** El encargado accede a la opción de gestionar la configuración, aparecen todos los servicios listados con sus diferentes propiedades, el encargado las modifica según considere y las guarda. El sistema pide al encargado que confirme que quiere modificar la configuración, y modifica los servicios actuales segun la nueva configuración.
   - **Secuencia excepción:** El encargado accede a la opción de gestionar la configuración, aparecen todos los servicios listados con sus diferentes propiedades, el encargado las modifica según considere pero sale sin guardar. El sistema deja los servicios sin modificar.
   - **Frecuencia esperada:** Baja, alrededor de una vez a la semana.
   - **Importancia:** Media
   - **Prioridad:** Alta
2. **Identificador:** CU2.Controlar el estado del sistema y los servicios
   - **Descripción:** El encargado ha de ser capaz de ver en todo momento en que estado de la ejecución esta cada servicio y si alguno de ellos ha generado errores y que errores son. En caso de encontrar errores, se tiene que registrar y notificar por correo electrónico.
   - **Alcance:** Empieza en el momento que algún servicio esta en funcionamiento y funciona mientras se estén ejecutando servicios.
   - **Nivel:** Tarea secundaria
   - **Actor principal:** Encargado

- **Actores secundarios:** Ninguno
- **Relaciones:** CU3, CU4, CU5, CU6, CU7
- **Precondición:** Algún servicio tiene que estar activo
- **Condición fin éxito:** El encargado del sistema es capaz de ver el estado y los posibles errores.
- **Condición fin fracaso:** El estado de algún servicio no se muestra correctamente o algún error no se ha registrado.
- **Secuencia normal:** El encargado comprueba la vista de los servicios en ejecución y ve que uno de ellos tiene una advertencia, accede a la advertencia y comprueba de que se trata. Una vez la resuelve, la marca como vista y el sistema deja de mostrarla.
- **Secuencia excepción:** El encargado comprueba la vista de los servicios en ejecución y ve que uno de ellos tiene una advertencia, accede a la advertencia y comprueba de que se trata. El encargado decide no resolver la incidencia en ese momento y la deja como estaba. El sistema sigue mostrando la advertencia.
- **Frecuencia esperada:** Se va a consultar diariamente
- **Importancia:** Máxima
- **Prioridad:** Alta

3. **Identificador:** CU3.Trasmisión del maestro de artículos

   - **Descripción:** El sistema debe transmitir la información con todos los artículos que tiene AX registrados(Maestro de artículos) a SEGA para que así este puede registrar los nuevos artículos que se den de alta. Esta acción se debe realizar diariamente durante la noche.
   - **Alcance:** Empieza cuando llegue la hora del dia asignada y termina una vez que se ha generado el fichero correctamente.
   - **Nivel:** Tarea principal
   - **Actor principal:** SEGA, AX
   - **Actores secundarios:** Ninguno
   - **Relaciones:** CU1, CU2
   - **Precondición:** El servicio debe estar encendido y sin errores graves.
   - **Condición fin éxito:** El fichero de comunicación se crea y se deposita correctamente, y SEGA lo reconoce como correcto.
   - **Condición fin fracaso:** Durante la ejecución se produce algún error que impide crear el fichero con la información.
   - **Secuencia normal:** El sistema obtiene la información que necesita de la base de datos de AX y la vuelca en un fichero de texto siguiendo el protocolo de comunicación, luego ese fichero se deja en el directorio de entrada de SEGA, donde este lo recoge satisfactoriamente y se archiva una copia para consultas posteriores.
   - **Secuencia excepción:** Durante el transcurso de la tarea del servicio, se produce una excepción, el servicio notifica ese error y, en caso de ser un error grave, se detiene el servicio. Si solo se trata de una advertencia, el servicio salta esta tarea y espera a que llegue el momento de la siguiente ejecución. Si parte del fichero se ha llegado a generar, se archiva como erroneo para consultar posteriormente.

- **Frecuencia esperada:** Se va a consultar diariamente
- **Importancia:** Máxima
- **Prioridad:** Alta

4. **Identificador:** CU4.Trasmitir los movimietos de artículos.
    - **Descripción:** El sistema debe transmitir los últimos movimientos de artículos (entradas o salidas) que se hayan producido en el almacén de SEGA y que afecten al stock que se tiene.
    - **Alcance:** Empieza cuando haya trascurrido el tiempo asignado desde la última ejecución y termina una vez se haya modificado correctamente el estos en AX.
    - **Nivel:** Tarea principal
    - **Actor principal:** SEGA, AX
    - **Actores secundarios:** Ninguno
    - **Relaciones:** CU1, CU2
    - **Precondición:** El servicio debe estar encendido y sin errores graves.
    - **Condición fin éxito:** El stock de AX se modifica y AX manda la respuesta de confirmación.
    - **Condición fin fracaso:** Durante la ejecución se produce algún error que impide que el stock se modifique correctamente.
    - **Secuencia normal:** El sistema obtiene la información que necesita del fichero de comunicación que ha generado SEGA, esa información se la trasmite a AX a través de su servicio web y espera su respuesta, una vez llega la respuesta afirmativa se archiva el fichero y se termina la ejecución.
    - **Secuencia excepción:** El sistema obtiene la información que necesita del fichero de comunicación que ha generado SEGA, esa información se la trasmite a AX a través de su servicio web y espera su respuesta. La respuesta llega con un error que se ha producido durante el cambio de stock en AX. Se genera un error, se notifica al encargado y se archiva el mensaje como erróneo para que el encargado pueda revisarlo posteriormente.
    - **Frecuencia esperada:** Aproximadamente cada 15 minutos.
    - **Importancia:** Media
    - **Prioridad:** Media

5. **Identificador:** CU5.Trasmitir el stock actual.
    - **Descripción:** El sistema debe transmitir durante la noche los artículos que tiene en stock y las cantidades de estos en el almacén de SEGA para que se compare con el de AX, así los dos sistemas estén coordinados.
    - **Alcance:** Empieza cuando llegue la hora del día asignada y termina una vez que se ha generado el fichero correctamente.
    - **Nivel:** Tarea principal
    - **Actor principal:** SEGA, AX
    - **Actores secundarios:** Ninguno
    - **Relaciones:** CU1, CU2

- **Precondición:** El servicio debe estar encendido y sin errores graves.
- **Condición fin éxito:** El stock de AX se modifica y AX manda la respuesta de confirmación.
- **Condición fin fracaso:** Durante la ejecución se produce algún error que impide que el stock se modifique correctamente.
- **Secuencia normal:** El sistema obtiene la información que necesita del fichero de comunicación que ha generado SEGA, con esa información busca las diferencias que tengan ambos sistema y , a través del servicio web, hace que AX modifique su stock hasta que los dos sistemas estén acorde. El sistema espera la respuesta de AX , una vez llega la respuesta afirmativa se archiva el fichero y se termina la ejecución.
- **Secuencia excepción:** El sistema obtiene la información que necesita del fichero de comunicación que ha generado SEGA, con esa información busca las diferencias que tengan ambos sistema y , a través del servicio web, hace que AX modifique su stock hasta que los dos sistemas estén acorde. El sistema espera la respuesta de AX La respuesta llega con un error que se ha producido durante el cambio de stock. Se genera un error, se notifica al encargado y se archiva el mensaje como erróneo para que el encargado pueda revisarlo posteriormente.
- **Frecuencia esperada:** Aproximadamente cada 15 minutos.
- **Importancia:** Máxima
- **Prioridad:** Alta

6. **Identificador:** CU6.Trasmitir los envíos entrantes al almacén.

   - **Descripción:** El sistema debe avisar a SEGA de que ha salido un camión con destino de dejar productos en su almacén. El sistema también debe recoger la confirmación de SEGA cuando la mercancía ha llega para avisar a AX que el envió se ha completado correctamente.
   - **Alcance:** Empieza cuando se realiza un envío con destino el almacén de SEGA y termina cuando SEGA confirma que se ha recibido la mercancía y se notifica a AX.
   - **Nivel:** Tarea principal
   - **Actor principal:** SEGA, AX
   - **Actores secundarios:** Ninguno
   - **Relaciones:** CU1, CU2
   - **Precondición:** El servicio debe estar encendido y sin errores graves.
   - **Condición fin éxito:** Se confirma que el envío ha llegado bien tanto en AX como en SEGA.
   - **Condición fin fracaso:** Durante la ejecución se produce algún error que impide que el transporte se confirme que ha llegado correctamente.
   - **Secuencia normal:** El sistema recoge la información del envío generado en AX y lo transmite a SEGA mediante el fichero de comunicación, SEGA recoge ese fichero y prepara su recibo. Posteriormente cuando el envío llega al almacén, SEGA genera un fichero de confirmación, que se lee y se notifica a AX para que este cierre el envío como satisfactorio y los ficheros de comunicación se archivan.

- **Secuencia excepción:** El sistema recoge la información del envío generado en AX y lo transmite a SEGA mediante el fichero de comunicación, SEGA recoge ese fichero y prepara su recibo. Posteriormente cuando el envió llega al almacén, SEGA genera un fichero indicando que el envío ha llegado con errores. Eso se notifica a AX para que deje el envió en estado erróneo y se notifica al encargado, además, se archivan los ficheros como erróneos.
- **Frecuencia esperada:** Aproximadamente cada 15 minutos minutos.
- **Importancia:** Máxima
- **Prioridad:** Alta

7. **Identificador:** CU7.TrasmitirEnviosSalientesAlmacen.

   - **Descripción:** El sistema debe avisar a SEGA de que un envio se tene que preparar y luego avisar a AX de que un envio ha sido preparado y se tiene que prepara su albarán de envio.
   - **Alcance:** Empieza cuando se programa un envio de productos y acaba cuando se realiza el albarán y se expede la mercancía.
   - **Nivel:** Tarea principal
   - **Actor principal:** SEGA, AX
   - **Actores secundarios:** Ninguno
   - **Relaciones:** CU1, CU2
   - **Precondición:** El servicio debe estar encendido y sin errores graves.
   - **Condición fin éxito:** El envio es expedido del almacén y se cierra la operación.
   - **Condición fin fracaso:** El envio no se puede preparar por falta de productos o falta información para poder realizar el albarán de envio.
   - **Secuencia normal:** El sistema busca en la base de datos si hay algún envio pendiente, si lo hay se lo hay se lo notifica a SEGA a través de un fichero de comunicación para que este empiece a preparar la comanda. Posteriormente cuando se haya terminado de prepara se notifica al sistema a través de un fichero, y el sistema se lo notifica a AX para que prepare el albarán de envio para que la comanda se pueda expedir.
   - **Secuencia excepción:** El sistema busca en la base de datos si hay algún envio pendiente, si lo hay se lo hay se lo notifica a SEGA a través de un fichero de comunicación para que este empiece a preparar la comanda. Posteriormente, SEGA informa de que no hay suficientes productos para preparar la comanda y se cancela su preparación. El sistema es informado de esta cancelación y este informa a AX para que tome medidas al respecto.
   - **Frecuencia esperada:** Aproximadamente cada 15 minutos.
   - **Importancia:** Máxima
   - **Prioridad:** Alta

3.2. Análisis del sistema

En esta sección se va a realizar el análisis del sistema a desarrollar. Primeramente se hablará sobre las tecnologías utilizadas y las ventajas que aporta su elección. Después, se presentara una

primera versión de la arquitectura del sistema mediante un diagrama de clases y se comentarán las decisiones tomadas para resolver los requisitos del sistema.

3.2.1. Tecnologías utilizadas

Las tecnologías que se utilizarán para el desarrollo de este proyecto se tuvieron claras desde el primer momento, principalmente por imposición de la empresa, ya que siempre han intentado utilizar las mismas tecnologías para sus proyectos. Por tanto, las tecnologías utilizadas y sus ventajas son las siguientes:

- Tanto el desarrollo de la pasarela de comunicación como de los servicios que implementa se realizaran sobre el framework de Microsoft .NET en su versión 4.6.1. Este framework utiliza el lenguaje de programación C# y permite un desarrollo de aplicaciones de escritorio para el sistema operativo de Windows rápido y sencillo. Mas concretamente, se realizará el sistema como una aplicación de Windows Forms, que permite realizar interfaces de usuario fácilmente y con unos resultados más que suficientes para los requisitos del sistema.

- El desarrollo del código se va a realizar usando el IDE de programación de Visual Studio 2017. Este entorno de desarrollo es el más recomendado para realizar aplicaciones en .NET, ya que también esta desarrollado por Microsoft y logra una integración completa con el framework. Además Visual Studio también aporta muchas herramientas para el desarrollo de interfaces gráficas para el sistema operativo de Windows. Finalmente, como el ERP de Dynamics AX con el que se conecta el sistema también esta desarrollado por Microsoft, Visual Studio permite acoplar ambos sistemas de manera cómoda.

- En la empresa se utiliza el gestor de base de datos de SQL Server, por tanto, para acceder a la base de datos se utiliza el programa de SQL Server Manager Studio 2014. Dado el gestor utilizado, este era el programa mas adecuado. Este gestor también esta mantenido por Microsoft, por tanto, tiene muchas ventajas dado que los demás sistemas se mantienen por la misma empresa.

- Parte de la funcionalidad del sistema se tiene que programar dentro del sistema de Dynamics AX, principalmente dentro de un servicio web. Para desarrollar este servicio web se utiliza una cuenta de desarrollo para acceder al sistema y se programa directamente en el propio editor de AX. El lenguaje de programación que utiliza es X++. Como este lenguaje también es propiedad de Microsoft, tiene facilidades para integrarse con C#.

3.2.2. Estructura del sistema

Teniendo en cuenta los requisitos descritos anteriormente y las casos de uso del sistema, se ha diseñado una primera versión de la arquitectura del programa, que se muestra en el diagrama de clases de la Figura 3.2. En esta primera versión ya se han tomado algunas decisiones importantes sobre el diseño, que se comentan a continuación:

- Por una parte, en la estructura general del sistema, como un requisito es que fuera lo mas modular y genérica posible para poder ampliar su funcionalidad en un futuro, se decidió por diseñar un controlador que se encargue de agrupar las diferentes tareas y mandar su ejecución cuando sea necesario. Este controlador también se encargaría de configurar las tareas al arrancar el programa y de apagar o encender los servicios que ejecutan dichas tareas. Aunque no se aprecia en el diagrama, cada tarea debe ejecutarse en una hebra diferente al controlador para lograr paralelismo, por tanto, el controlador también deberá encargarse de actualizar la vista dado que las otras hebras no pueden hacerlo. También hay que comentar que desde el punto de vista del controlador, las diferentes tareas se ven como un mismo tipo genérico.

- Por lo que refiere a las comunicaciones del sistema con otros sistemas externos, se han detectado tres posibilidades diferentes, y por tanto, se ha diseñado una clase para cada tipo de comunicación. Por un lado tenemos la comunicación con la base de datos de AX. Esta conexión seguro requerirá de algún tipo de información de conexión o algún driver. Por otro lado tenemos la conexión con el servicio web de AX. Esta conexión seguro que también requiere de algún tipo de información de conexión, además de una clase específica de mensaje para que se entiendan ambos sistemas. Finalmente, se han considerado los directorios en los que se depositan o recogen los ficheros de SEGA como una comunicación con un sistema externo, por tanto, también se ha diseñado una clase específica para ello.

- En cuanto al modelo del sistema, en un principio no parece ser necesarias muchas clases. Por el momento solo se encontraron necesarias una clase para que el sistema se entienda con el servicio web, y otra para mantener la información de los posibles errores o advertencias encontradas durante la ejecución de las tareas.

- Finalmente, el módulo de las tareas, donde estará concentrada la mayoría de la funcionalidad del sistema. Primeramente, para mantener la abstracción del sistema, se diseñó una tarea abstracta que sirva de base para todas las demás posibles tareas del sistema. Se pensó que toda tarea debía tener un identificador único para poder diferenciarlas, una descripción que indicase de manera corta su función, un ciclo de ejecución para saber cada cuánto tiempo o a qué hora del día se tiene que ejecutar, un directorio base donde tendría los ficheros necesarios para su ejecución o los ficheros temporales que finalmente se moverían a su destino, y una lista con los errores o advertencias que ha podido ir generando. Además todas las tareas deberían tener también un método que contenga la funcionalidad y que se ejecute cuando sea el momento, y dos métodos que se ejecuten al iniciar el servicio y al apagarlo. Por lo que se refiere a las tareas concretas que extienden a la tarea genérica, se han detectado dos tipos diferentes. Por una parte tenemos las tareas que obtienen información de AX y la mandan a SEGA y por otra parte las que hacen lo contrario, obtiene información de SEGA y las la mandan a AX. Esta agrupación es solo a nivel conceptual, pero útil ya que permite agrupar las tareas en función de las comunicaciones que necesitan.

    - Las tareas de AX hacia SEGA necesitarán obtener la información de la base de datos, para luego generar el fichero de comunicación que depositarán en el directorio de SEGA.
    - Las tareas de SEGA hacia AX necesitaran recoger los ficheros de los directorios de SEGA, para luego descomponer la información y crear un mensaje de comunicación que se mandará a AX a través de su servicio web. Estas tareas dispondrían de una clase externa que les leería el fichero y les devolvería la información necesaria.

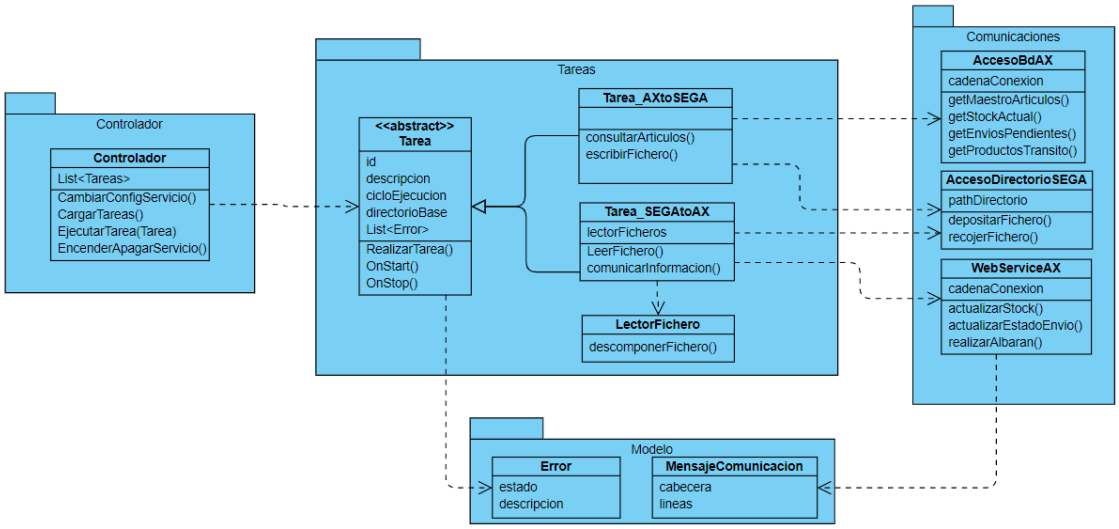


Figura 3.2: Diagrama de clases obtenido en la fase de análisis

Esta es una primera aproximación al sistema final, en la siguiente fase de diseño se va a completar la estructura con todas las tareas que se necesitan y los diferentes tipos de datos que se necesitan.

3.3. Diseño de la arquitectura del sistema

Partiendo del diagrama de clases obtenido en la fase de análisis anterior, se ha ampliado su definición en un nuevo diagrama de clases que se muestra en la Figura 3.3 y que incluye los tipos de datos de los atributos del sistema además de todas las clases necesarias para cumplir toda la funcionalidad de todos los casos de uso del sistema.

La estructura general del sistema sigue siendo la misma, solo que se han añadido más componentes para completar los requisitos del sistema. A continuación se van a explicar los diferentes añadidos de este nuevo diagrama y su explicación:

- Empezando por el modelo de datos, que se ha separado entre los demás componentes para lograr un diagrama mas legible, se han añadido más mensajes de comunicación con el servicio web ya que solo con uno no se podían englobar todas las necesidades. En concreto, se ha diseñado un mensaje para cambiar el stock de varios productos, para ello era necesario un mensaje de cabecera que contiene información común a todos las líneas, por ejemplo, el motivo del ajuste o la fecha, y también una lista de líneas donde se enumeran todos los productos y las cantidades a modificar. Los otros dos mensajes se utilizan para el caso de uso de realizar envíos de productos. En concreto uno es necesario para cerrar una ruta de picking, y el otro para realizar el albarán una vez se ha expedido la mercancía. Para terminar, también se han añadido dos enumeraciones, una para controlar el estado de las tareas para saber si han terminado o no y si han generado errores graves, y otra para saber el tipo de errores que ha generado alguna tarea.

- En los componentes de comunicaciones, se han añadido una interfaz para cada tipo de comunicación. Aunque estas interfaces no son estrictamente necesarias ya que los medios de comunicación no van a ser intercambiables ni extensibles para las tareas ya creadas, sí que facilitan en enorme medida el poder realizar test unitarios y de integración en las tareas, por eso se han añadido. Las diferentes tareas que lo necesiten referenciarán estas interfaces a través de la inyección de dependencias por constructor.
- En cuanto a las tareas del sistema, se han añadido todas las necesarias para cumplir con los requisitos del sistema, a continuación se va a explicar cada una de ellas que misión cumplen:
  - Por una parte, la tarea que queda sin agrupar llamada Tarea_Wathchdog sirve para mantener este mismo sistema de watchdog. En concreto se encarga de crear regularmente un fichero de texto vacío en un directorio para que el sistema del watchdog ajeno a este sepa que no se ha colgado. También comprueba la lista de procesos del sistema y, en caso de que no este ejecutándose, ejecuta el watchdog. Esta funcionalidad no se consideró que se utilizara como una tarea mas del sistema en la fase de diseño, pero al profundizar mas en su funcionamiento se vió que es muy fácil adaptarlo al funcionamiento de las tareas.
  - La tarea llamada Tarea_CambioInventario cumple con la funcionalidad del caso de uso CU4.Trasmitir los movimientos de artículos.
  - La tarea llamada Tarea_AcualizarStock cumple con la funcionalidad del caso de uso CU5.Trasmitir el stock actual.
  - Las tareas llamada Tarea_ConfirmacionEntrada y Tarea_MovimientosAlmacen cumplen con la funcionalidad del caso de uso CU6.Trasmitir los envíos entrantes al almacén. Como esta funcionalidad requiere de paso de información en los dos sentidos, es más sencillo partir la funcionaldad en dos tareas distintas, una para cada sentido.
  - Las tareas llamada Tarea_ConfirmacionSalida y Tarea_EnviosPendiente cumplen con la funcionalidad del caso de uso CU7.TrasmitirEnviosSalientesAlmacen. Al igual que con el caso de uso anterior, esta requiere de traspaso de información en los dos sentidos y por eso también se divide en dos tareas.
  - Finalmente, la tarea llamada Tarea_MaestroArticulos cumplen con la funcionalidad del caso de uso CU3.TransmitirMaestroArtículos.

3.4. Diseño de las secuencias del sistema

Para completar el diseño del sistema y mostrar una visión mas detallada de las diferentes secuencias y pasos de información del sistema, se van a mostrar una serie de diagramas de secuencia UML que escenifican las diferentes tareas que ha de cumplir el sistema.

La secuencia de cambios de inventario se ve en la figura 3.4 muestra cómo se produce el paso de información de SEGA a AX de los últimos movimientos de stock del amacén.

La secuencia de confirmación de entrada se ve en la figura 3.5 muestra como se informa a AX de que una mercancia ha llegado correctamente al almacén de SEGA.

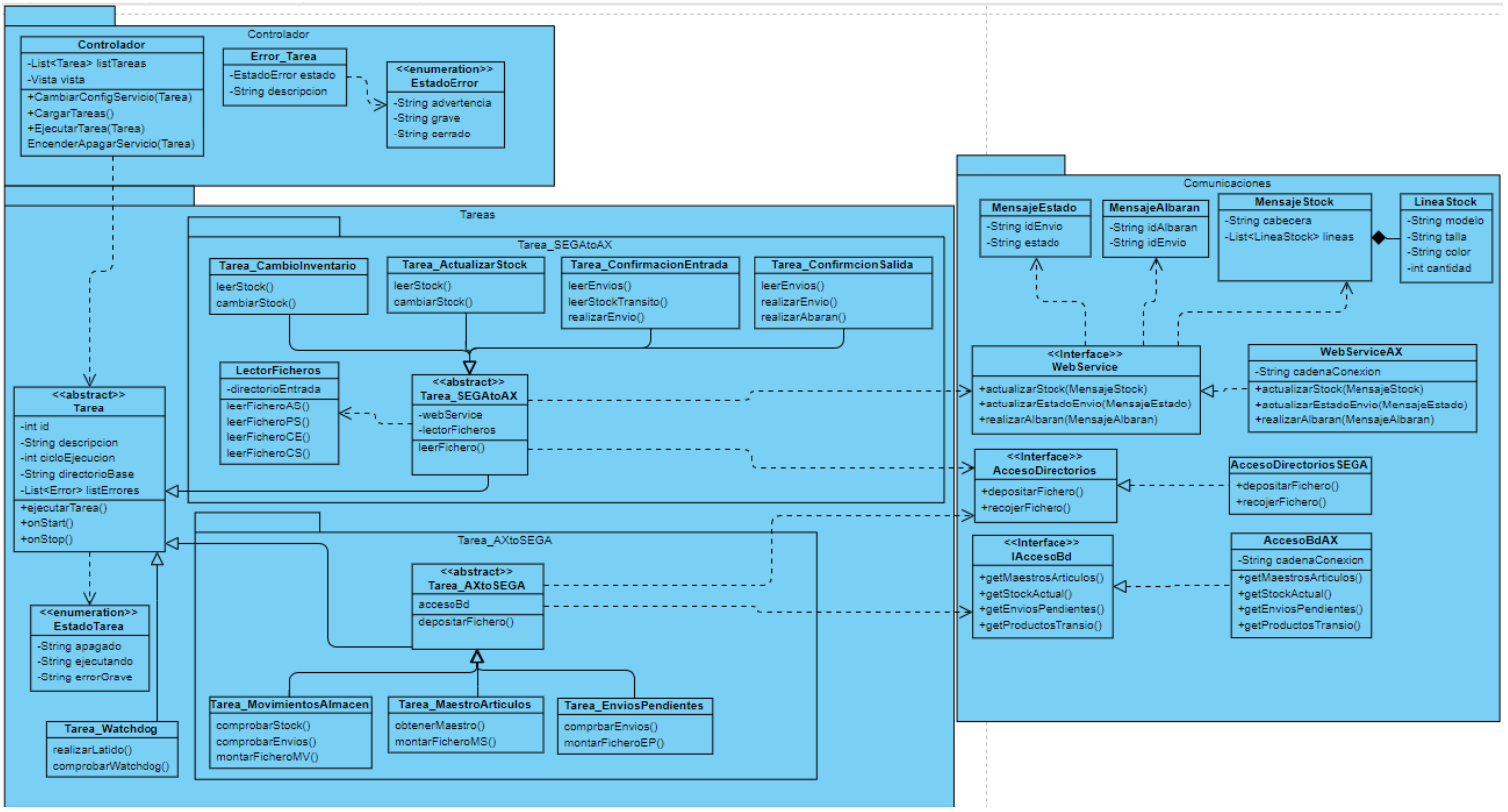


Figura 3.3: Diagrama de clases obtenido en la fase de diseño

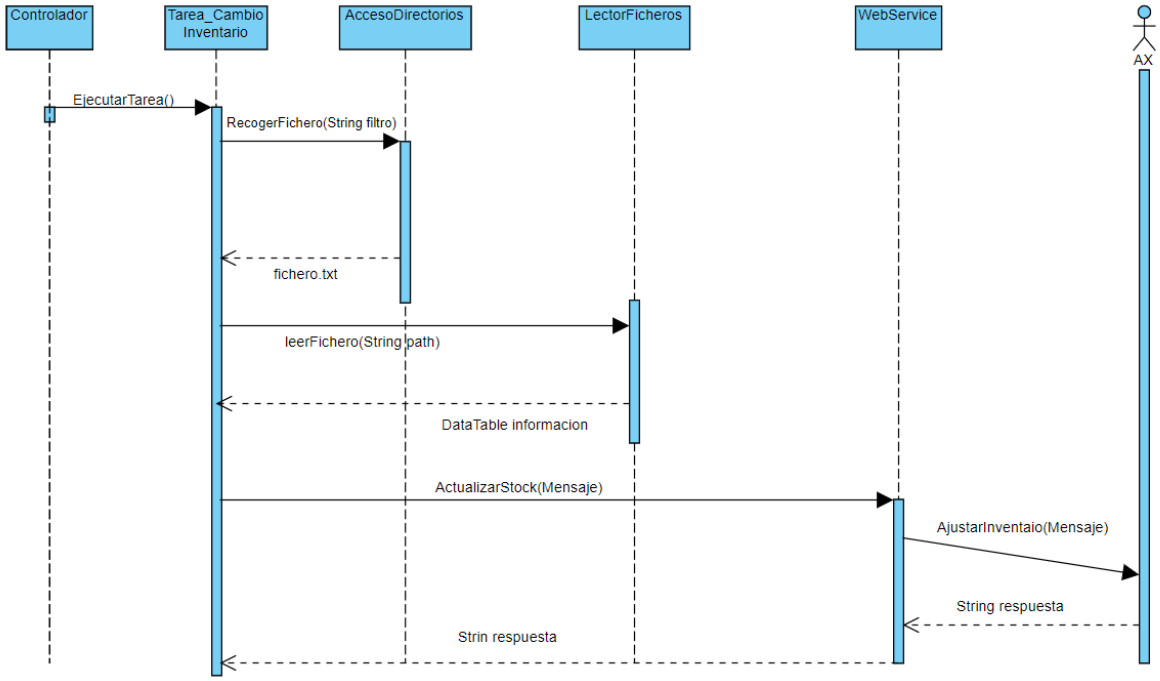


Figura 3.4: Diagrama de la secuencia de cambio de inventario

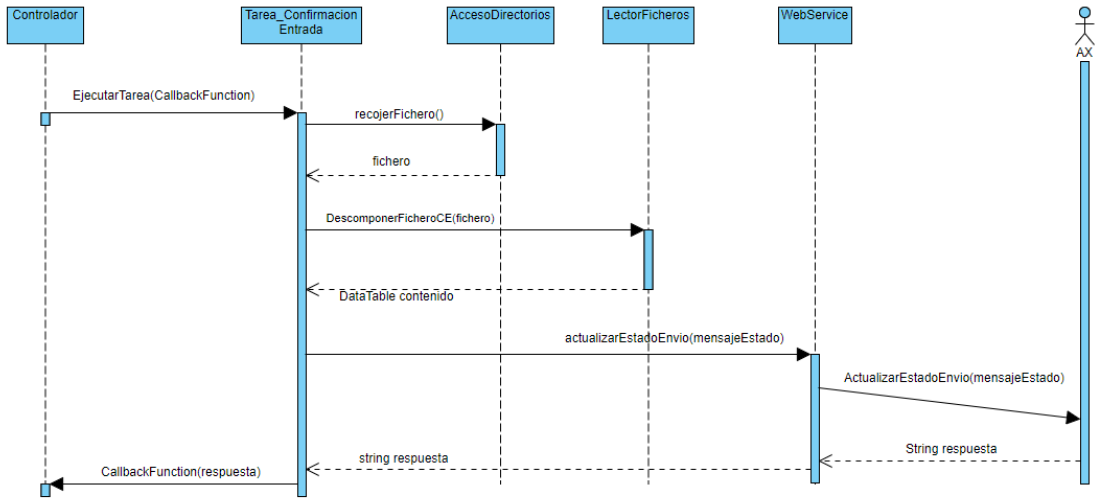


Figura 3.5: Diagrama de la secuencia de confirmación de entrada al almacen

La secuencia de confirmación de salida que se ve en la figura 3.6 muestra cómo se informa a AX de que una mercancía ha sido correctamente expedida del almacén de SEGA para que realize el albarán de envío.

La secuencia de diferencia de stock que se ve en la figura 3.7 muestra cómo se informa a AX del stock actual de SEGA para que ajuste el stock y así ambos sistemas estén coherentes.

La secuencia de los maestros de artículos que se ve en la figura 3.8 muestra como se manda a SEGA el maestro de artículos de AX para que registre los nuevos artículos

La secuencia de movimientos entre almacenes que se ve en la figura 3.9 muestra cómo AX avisa a SEGA de que un envío de productos va de camino a su almacén.

La secuencia de envíos pendientes que se ve en la figura 3.10 muestra como AX avisa a SEGA de que tiene algunos envíos pendientes que tiene que preparar para su expedición.

De todas las secuencias se puede destacar que todas se inician cuando el controlador llama al método RealizarTarea pasando como parámetro una función de callback. Dado que las tareas se ejecutan en hebras distintas, no pueden actualizar la interfaz gráfica directamente, y además estar comprobando por parte del controlador constantemente si alguna tarea ha terminado no sería eficiente. Por tanto, se optó por diseñar un mecanismo de observador-observable como este. Así pues, cuando una tarea termina su ejecución, utiliza el método que recibió como parámetro para avisar de que ha terminado y cual ha sido el resultado. El controlador recibe esa información y actualiza la vista según corresponda.

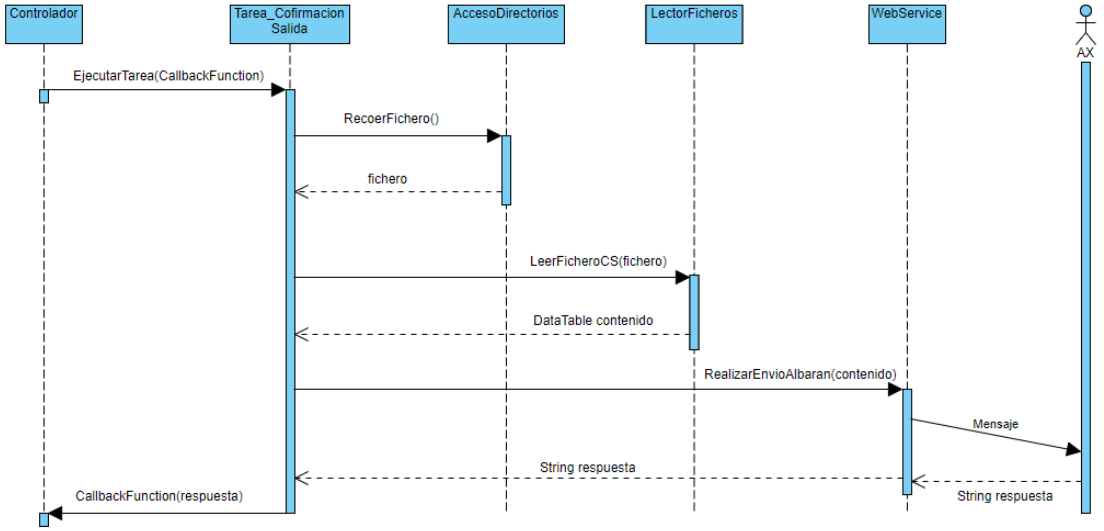


Figura 3.6: Diagrama de la secuencia de confirmación de salida del almacen

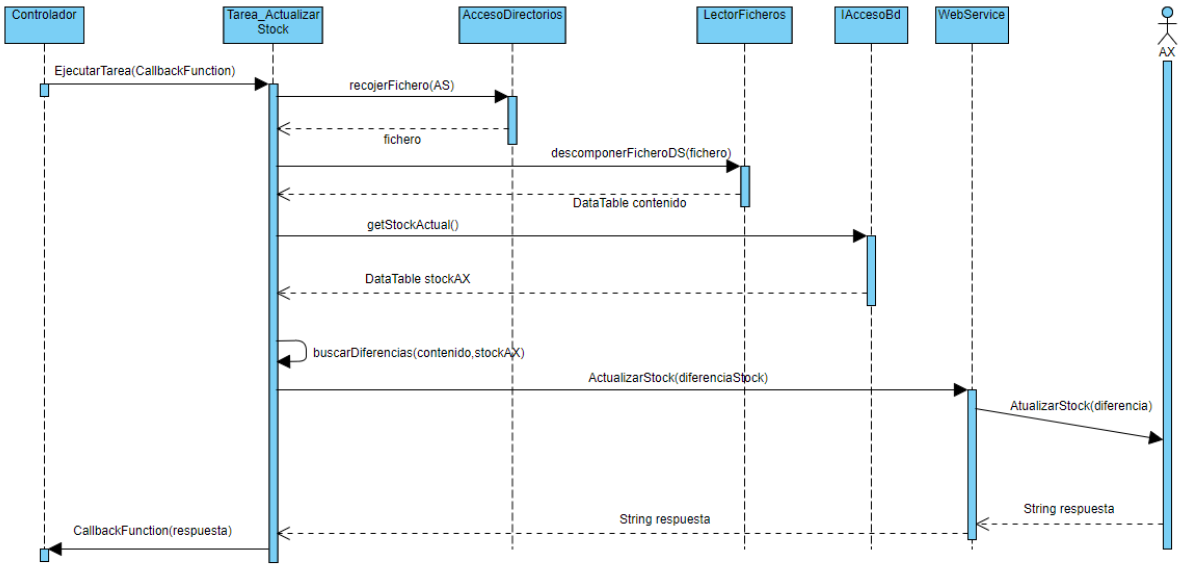


Figura 3.7: Diagrama de la secuencia de diferencia de stock

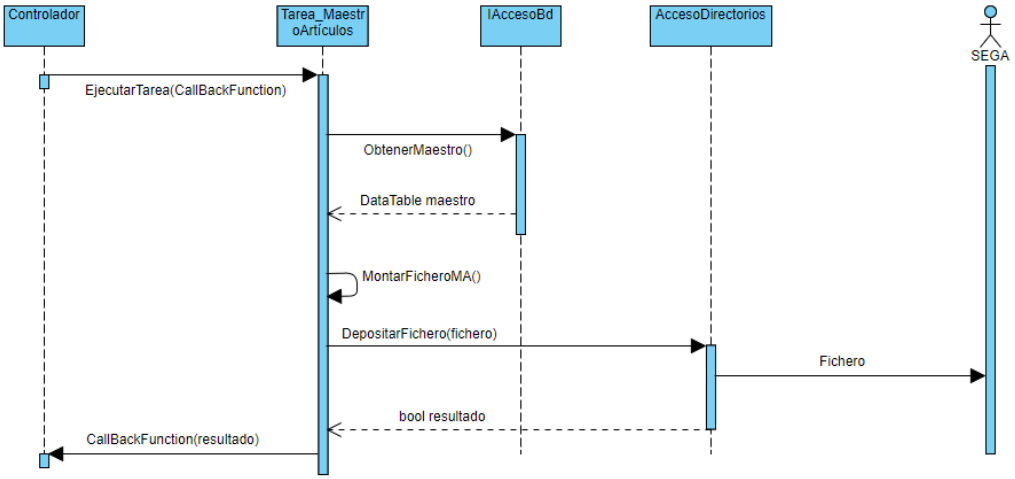


Figura 3.8: Diagrama de la secuencia del maestro de artículos

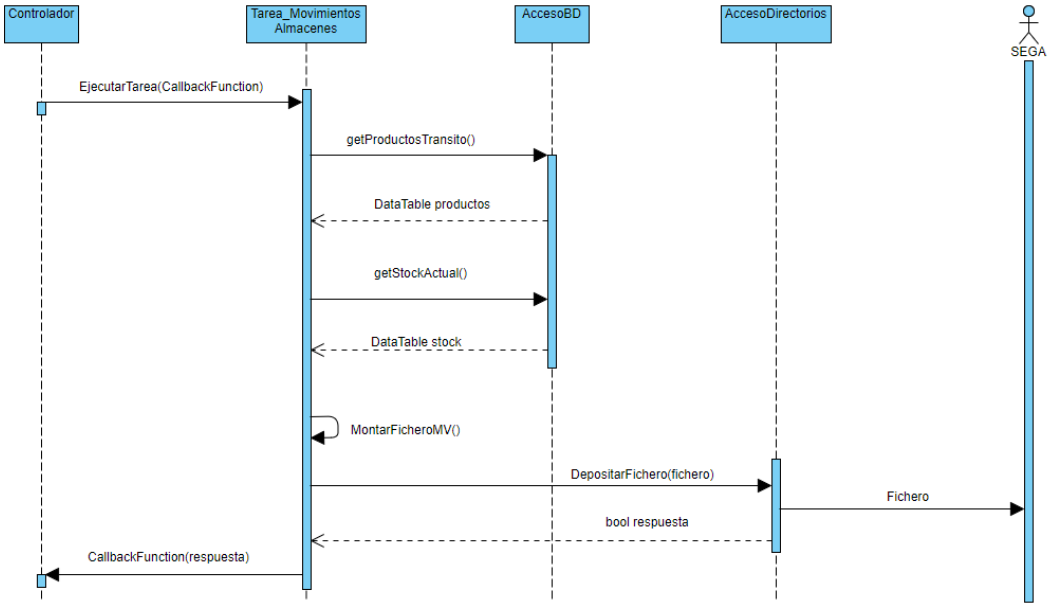


Figura 3.9: Diagrama de la secuencia de los movimientos entre almacenes

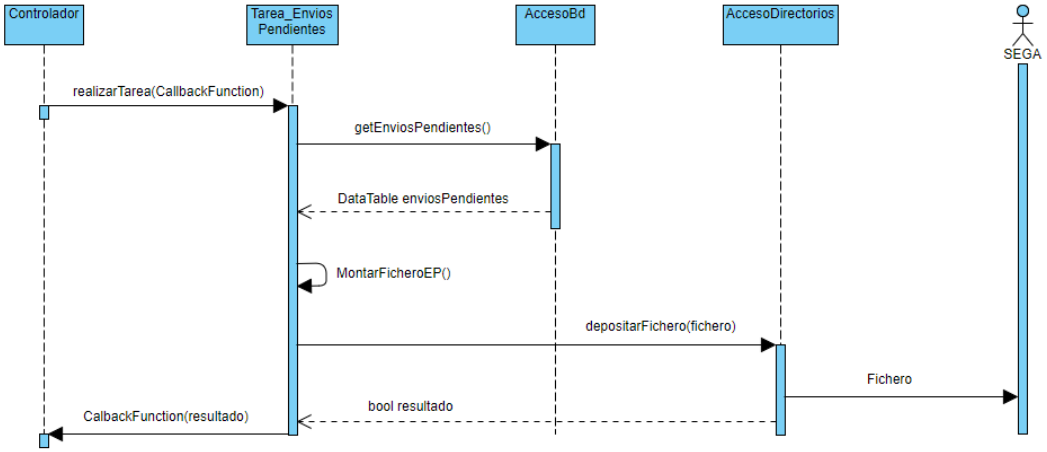


Figura 3.10: Diagrama de la secuencia de los envíos pendientes

3.5. Diseño de la interfaz

En esta sección se van a comentar las decisiones de diseño tomadas para las diferentes vistas del sistema. También se va a mostrar un prototipo de las diferentes pantallas principales.

En la Figura 3.11 podemos ver la pantalla principal del sistema. Esta es la pantalla más importante dado que será la que estará en primer plano la mayor parte del tiempo, por tanto, es donde recaen la mayor parte de decisiones de diseño.

Como el sistema va a ser utilizado principalmente por personal informático, y la mayoría del tiempo funcionará sin que ningún usuario este presente, el sistema no tiene una interfaz demasiado vistosa, más bien se ha intentado buscar su utilidad.

El objetivo principal de esta pantalla principal es que de un vistazo rápido sea fácil conocer el estado del sistema, es decir, cuántos servicios estan encendidos y cuántos no, además de si alguno de ellos tiene errores. Es por eso que se decidió diseñar el sistema con tonos grises neutros, menos por los indicadores de la ejecución de los servicios y los indicadores de errores que tienen colores mas llamativos.

También se intentó que desde la pantalla principal estuvieran disponibles todas las acciones relacionadas con los servicios. Por eso desde la lista de servicios del panel central se pueden realizar las acciones encender o apagar un servicio, forzar la ejecución de un servicio y acceder a los ficheros que utiliza dicho servicio. Para cambiar la configuración de los diferentes servicios sí que se diseñó otra pantalla que se puede acceder desde el menú superior y que se puede ver en la figura 3.12.

También era necesario, por requisito de la empresa, que hubiera un log en tiempo real en esta pantalla principal que mostrara los avances de la ejecución de las diferentes tareas. Este log es separado del log de incidencias que se mostrará después. En este log principal solo aparecen mensajes informativos, que principalmente sirven para conocer la ejecución de los servicios en funcionamiento.

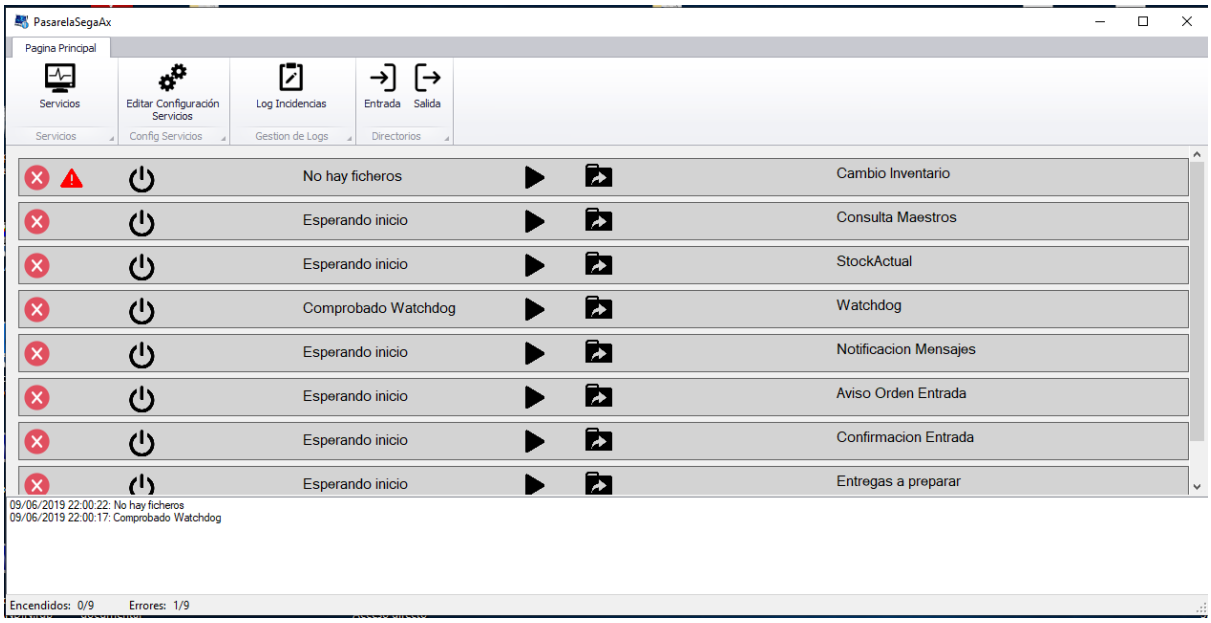


Figura 3.11: Interfaz principal del sistema donde se muestran los servicios en ejecución y su estado

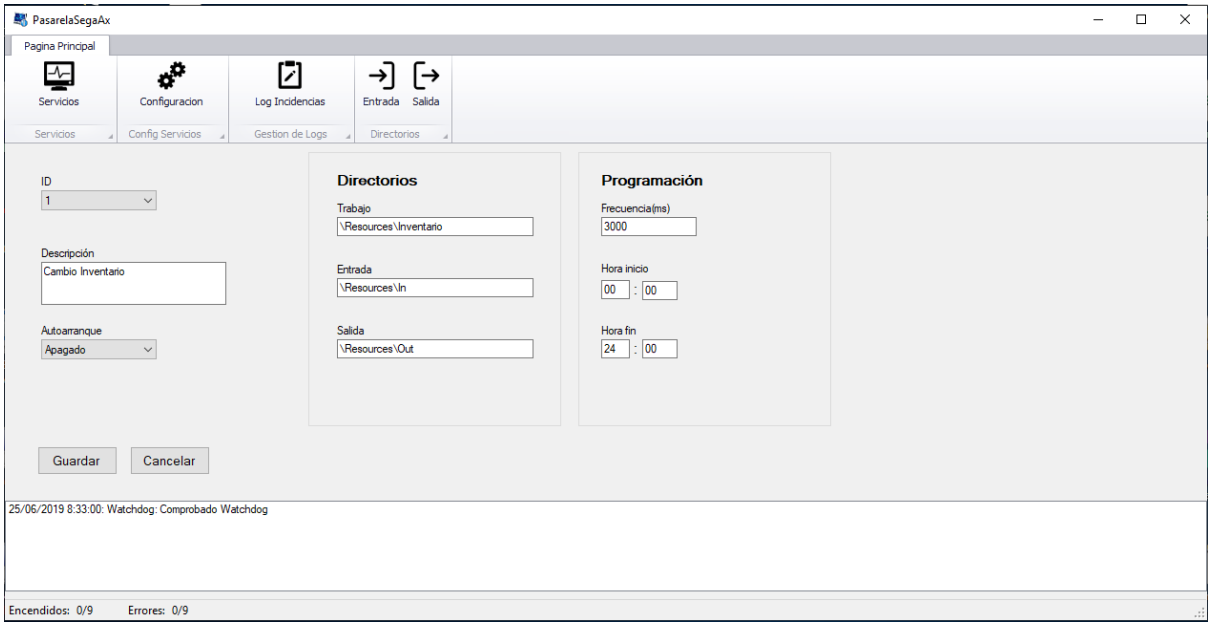


Figura 3.12: Interfaz que muestra el cambio de configuración de las tareas

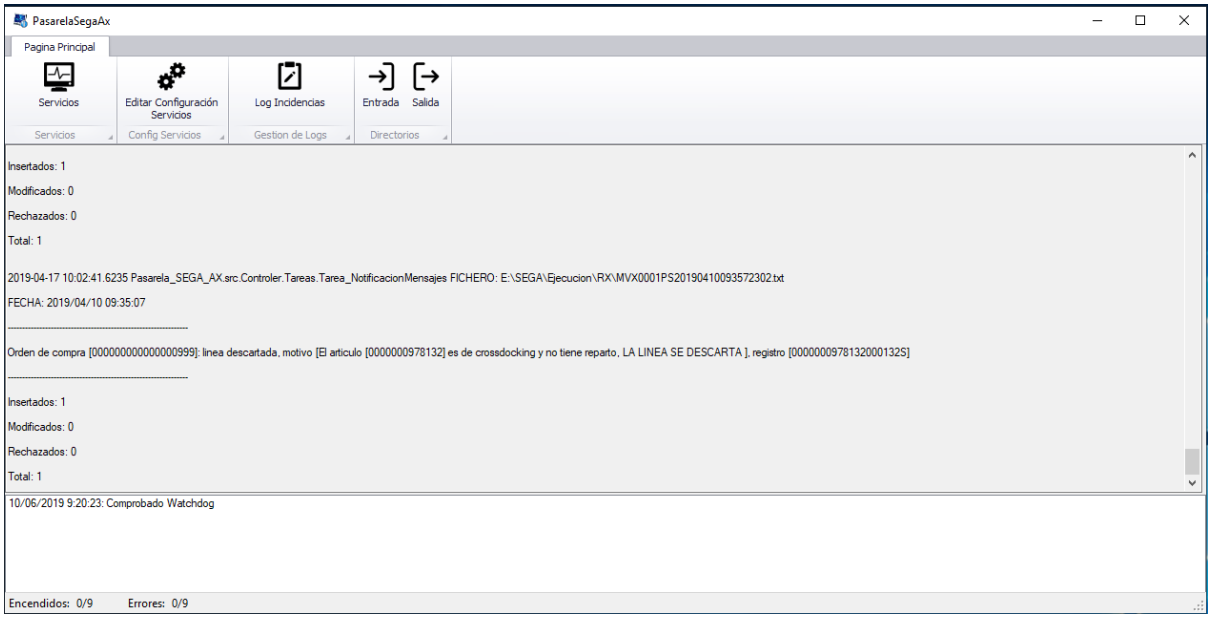


Figura 3.13: Interfaz que muestra el log con el registro de las incidencias del sistema

Finalmente, el log de incidencias que se muestra en la Figura 3.13, al que se puede acceder desde el menú superior, y que registra las incidencias que ha ido generado el sistema, con la fecha y hora de su aparición, y una traza del problema. Estas incidencias registradas están pensadas para ser entendidas por el personal informático de la empresa, por tanto suelen ser bastante técnicas. En el diseño de esta pantalla no hay nada que destacar dado que solo es necesaria la lectura de estas incidencias.

Capítulo 4

Implementación y pruebas

En este capítulo se va a relatar como se ha llegado a implementar el sistema partiendo del diseño y el análisis del sistema obtenidos en el apartado anterior. Con ello se comentarán las decisiones tomadas respecto a la implementación. Finalmente se mostrarán las pruebas de validación y verificación diseñadas e implementadas para asegurar un sistema de la mayor calidad posible.

4.1. Detalles de implementación

Para la implementación de este sistema, se tomó la decisión de dividir todo el desarrollo en tres fases. Primeramente se desarrollaría la base de la pasarela, de modo que permitiera programar tareas genéricas las cuales se ejecutarían cada cierto tiempo o a cierta hora del día, además de gestionar su configuración y ejecución. Luego, usando las la pasarela como base, se desarrollarían todas las tareas necesarias para poder cumplir con la funcionalidad del sistema y finalmente, se desarrollaría la parte del servicio web, dentro del sistema de AX, necesario para poder comunicarse. En los siguientes apartados se va a dar mas detalles de la implementación de cada sección y el porque de ese orden.

4.1.1. Implementación de la pasarela

El primer bloque a desarrollar sera la base del programa que permita programar tareas mediante servicios que se ejecuten periódicamente. La razón para desarrollar este bloque con independencia de las diferentes tareas que se van a desarrollar en los siguientes bloques, es para favorecer la abstracción del código y lograr un sistema que sea mas fácilmente ampliable en el futuro.

La interfaz gráfica del sistema también se va a desarrollar en este apartado, dado que las tareas no tienen ninguna interfaz en específico y el servicio web está en otro sistema, tiene más sentido implementar las diferentes interfaces en este momento.

Por lo que refiere al sistema de *watchdog*, aunque se implemente como una tarea más del sistema, su funcionalidad esta relacionada con el sistema base, por tanto, también se desarrollará en esta fase.

- **Configuración del las tareas en el arranque.** Nada más se inicia el sistema tiene lugar el proceso de arrancar los diferentes servicios del sistema. Este proceso tiene varios pasos. Primero el sistema lee la información de la configuración de las tareas de un fichero xml. Esta configuración indica toda la información necesaria para configurar las tareas. Después, según el identificador de cada tarea, el sistema instancia un tipo de tarea u otra. Esta instancia no está directamente en el código del controlador, sino que se obtiene la tarea ya instanciada a través de una clase Factory que instancia una tarea u otra en función del parámetro identificador. Finalmente, cuando ya se tienen las clases tarea instanciadas y configuradas, se añade su panel correspondiente a la vista del sistema y se guarda en una lista para su futura ejecución. También, en caso de que algún servicio este programado para que se ejecute directamente cuando arranque el sistema, este lo arranca.

- **El método de callback**. Cuando llega el momento de ejecutar una tarea, el sistema crea una nueva hebra de ejecución y manda ejecutar en ella la tarea concreta. La ejecución de la tarea en una hebra diferente a la principal aporta varias ventajas, como por ejemplo que el sistema no bloquee la interfaz gráfica a causa de operaciones costosas, o que se puedan procesar varias tareas simultáneamente, sin embargo, también tiene sus inconvenientes. Una hebra secundaria no puede modificar directamente la vista, además, también se necesita algún mecanismo especial para notificar cuando termina su ejecución. Para solucionar este problema se optó por utilizar un método de callback. Concretamente, cuando el programa principal llama al método *EjecutarTarea* de una tarea en concreto, le pasa como argumento un método delegado que toma como argumento una string y una clase. Cuando la tarea termine su ejecución o encuentre algo destacable que se deba notificar para actualizar a la vista, llamara al método callback que ha recibido como argumento. Este método callback se ejecutará en la hebra principal donde ya tiene acceso a todo lo necesario.

- **Programación de las tareas.** Para la ejecución cíclica de las tareas se utiliza un *timer*. Concretamente se utiliza un *timer* sin repetición que cuando llega a cero, ejecuta la tarea y programa el siguiente *timer*. Tanto las tareas que se ejecutan cada cierto tiempo como las tareas que se ejecutan a cierta hora del día utilizan el mismo tipo de *timer*. Para programar el timer, se comprueba el atributo que tiene la tarea, si se tiene que ejecutar cada cierto tiempo, se programa un nuevo *timer* que tarde ese tiempo en accionarse. Si se trata del tipo que se ejecuta a cierta hora del día, se comprueba cuanto tiempo hay de diferencia entre la hora actual y la hora de ejecución, y se programa el *timer* con la diferencia de tiempo, así se ejecutará a la hora correcta.

- **Funcionamiento del watchdog.** Para mantener el sistema en ejecución pese a bloques o cierres inesperado se utiliza un sistema de *watchdog*. En concreto, este sistema se ejecuta en un proceso a parte del sistema de pasarela para que los bloques no le afecten. El *watchdog* primero comprueba de la lista de procesos del sistema si hay algún proceso en ejecución con el nombre de la pasarela. Si hay más de uno cierra procesos para que solo quede uno, si no hay ninguno, lo ejecuta de nuevo. Este sistema sirve para controlar las caídas del sistema, pero podría darse el caso que el proceso de la pasarela se estuviera ejecutando pero estuviera congelado.

Para controlar los bloqueos del sistema de pasarela, se utiliza un sistema de *hearthbeat*. Concretamente, el sistema principal utiliza una de las tareas para crear periódicamente un fichero vacío en un directorio concreto. Por otra parte el sistema de *watchdog* busca si existe ese fichero y si lo encuentra lo borra. En caso de no encontrarlo, el *watchdog* supone que el sistema principal ha entrado en un estado de bloqueo y lo reinicia. La tarea del sistema principal que crea el fichero, también se encarga de comprobar la lista de procesos y ejecutar el *watchdog* en caso de que no esté ejecutándose. Esto es necesario dado que el *watchdog* también podría colgarse.

- El sistema de log. Para mantener el log del sistema se utiliza una biblioteca o *dll* de C# llamada NLog [2] que permite realizar esta función muy fácilmente. En concreto esta biblioteca se encarga de crear los ficheros que mantienen la información y añadir líneas en función de los mensajes que se quieran guardar. También se encarga de mandar un correo electrónico al encargado cuando se produce un error grave que deba ser notificado. Esta funcionalidad se utiliza principalmente en el método de callback nombrado anteriormente. Cuando este método recibe una llamada producida por una excepción en la ejecución de la tarea, la traza de esa excepción y un mensaje que lo resume se mandan a registrar en el log. En caso de que el error se considere de máxima gravedad, también se notifica mediante un correo electrónico.

4.1.2. Implementación de las tareas programadas

El segundo bloque a desarrollar serán las diferentes tareas que cumplan con los requisitos de comunicación del sistema. Estas tareas se programarán a partir de la funcionalidad aportada por la pasarela programada en el bloque anterior.

Las secuencias que diferencian cada tarea ya se han explicado en los apartados de análisis y diseño, por tanto, en este apartado se explicarán detalles de la implementación que, por lo general, son comunes a todas las tareas.

- Crear directorios. Todas las tareas utilizan directorios para crear los ficheros temporales que después rellenan con información y depositan, o para mover los ficheros que más tarde van a leer. La idea de utilizar directorios propios para cada tarea es para evitar posibles problemas de I/O y por mas comodidad al no tener que preocuparse por si otras tareas modifican los mismos ficheros. La idea es que estos directorios no se tengan que crear manualmente, sino que el sistema los cree la primera vez que los necesite. Por tanto, todas las tareas tienen un primer bloque de código nada mas empieza su ejecución en el cual comprueban si tienen todos los directorios que vayan a usar, en caso contrario, los crean.

- Paso de información en *DataTables*. Tanto la información que se extrae de los ficheros de comunicación, como la que se obtiene de la base de datos de Dynamics, se mueven utilizando *DataTables*. Las DataTables son una clase que tiene .NET y que permite transportar información de manera rápida sin la necesidad de crear tantas clases de modelo. En concreto estas tablas tienen una serie de columnas y unas filas que contienen información para esas columnas, similar a las tablas de una base de datos relacional. La ventaja de utilizar este mecanismo es que, como una misma tarea esta diseñada para leer un tipo

concreto de fichero o una tabla en concreto de la base de datos y eso no va a cambiar, utilizar DataTables agiliza mucho el desarrollo.

- Coger ficheros filtrando por nombre y antigüedad. Las tareas que necesitan leer los ficheros de comunicación mandados por SEGA no los recogen de manera aleatoria, sino que siguen un patrón. Según el protocolo de comunicación, cada mensaje esta identificado por su nombre, en concreto, tiene un valor numérico único seguido de dos caracteres que indican su tipo. Con esto, cada tarea busca entre todos los ficheros del directorio de entrada aquellos que en su nombre contengan los carácteres que lo identifiquen como el tipo de fichero que busca. En caso de haber mas de algún fichero, aunque esto es difícil que pase, se cogería aquel que tuviera una fecha de actualización mayor, es decir, el que llegó antes.

- Encontrar errores. Si durante el desarrollo alguna tarea encontrase algún error, esta tiene que notificárselo al controlador. Para la mayoría de errores que se generan es suficiente con rodear el código con secuencias *try/catch* y, de encontrar algún error, avisar al controlador con el método de callback. Sin embargo, los errores que se producen durante la ejecución en el servicio web funcionan diferente. Aunque los errores se produzcan durante la ejecución del servicio web, se tienen que notificar en la pasarela y no en el ERP. Esto se decidió así para agrupar todos los errores que estén relacionados con el sistema. Por tanto, de darse algún error en el servicio web, este contesta con un mensaje de error, el cual se recibe en la tarea de la pasarela y se notifica al controlador.

- Conexión síncrona con el web service. Cuando alguna tarea manda información al servicio web, se queda a la espera de que este le conteste. Esto es necesario porque en caso de producirse algún error se tiene que notificar y, en caso de error grave, detener el servicio. Esta conexión síncrona supondría un rendimiento bajo del sistema si no fuera porque cada tarea funciona en una hebra distinta. Este es uno de los motivos principales por los que se optó por ejecutar cada tarea en una hebra de ejecución.

4.1.3. Implementación del servicio web

El tercer y último bloque a desarrollar sera el servicio web que se encuentra en el sistema de AX y que es necesario para que este pueda recibir información de la pasarela. Aunque este servicio web no forme parte del sistema de pasarela, si que se va a comentar aquí su implementación ya que es necesaria para el funcionamiento correcto del sistema.

- *Data contracs.* Para el paso de información entre el servicio web y cualquier otro sistema que quiera comunicarse con él, se utilizan los denominados *Data Contracts*. Estos *Data Contracts* son clases de modelo que sirven como referencia tanto para el servicio web como para el otro sistema. De esta forma, la pasarela puede instanciar estas clases y pasarlas como argumento en las llamadas a los métodos del servicio web. Al utilizar ambos sistemas las mismas clases, es muy fácil pasar la información. La respuesta del servidor también se realiza utilizando estas mismas clases de *Data Contracts*. Todos los métodos del servicio web responden con una clase respuesta que encapsula el resultado de sus operaciones.

- *Invent journals* para cambiar stock. En los ERP el inventario se mantiene de forma especial. El total de stock de un producto se resume en el total de transacciones que se han realizado de ese producto, es decir, que el total de un producto es igual al total

de aumentos de stock de ese producto menos el total de disminuciones del mismo. Por tanto, para aumentar o disminuir el stock no se puede modificar directamente el stock del producto, sino que hay realizar lo que se llama un *Journal Trans*. Por ejemplo, cuando el sistema de pasarela llama al método del servicio web para aumentar el stock de un producto, el servicio web insertara una nueva fila en la tabla de la base de datos que contiene los *Journal Trans*. Al insertar esa transacción, el resumen del stock total aumenta consecuentemente.

- Transacciones. El sistema ha de tener algún mecanismo de persistencia para que, en caso de que el procedimiento de modificar el stock falle a mitad, no se produzca ningún desbarajuste en la consistencia del sistema. Para lograr esto, se utilizan transacciones en todos los procedimientos críticos, como es el caso de modificar el stock. Así, si durante la ejecución falla, todas las inserciones en la tabla de *Journal Trans* se descartan y el stock no se modifica.

4.2. Verificación y validación

En esta sección se van a comentar las medidas tomadas para comprobar que el sistema desarrollado cumple con los requisitos deseados y que además lo hace dentro de unos parámetros de calidad. También se comentarán los recursos que se han tenido para probar el sistema y los resultados logrados.

4.2.1. Entorno de pruebas

Para poder probar un sistema adecuadamente es muy importante disponer de un buen entorno de pruebas, en el cual se pueda experimentar libremente sin dañar a los otros sistemas en funcionamiento. En el caso de este proyecto, desde el lado de AX, se disponía de una copia de la base de datos real, en la cual se realiza un volcado de los datos reales cada semana. Esto ha permitido realizar todas las pruebas necesarias con las conexiones entre AX y la pasarela de forma muy cómoda. Además también ha permitido probar los métodos del servicio web libremente. Por otra parte, el funcionamiento de SEGA no se han podido probar realmente, ya que este sistema no tiene entorno de pruebas y se ve como una caja negra desde el sistema de pasarela. Por tanto, se supone que el sistema de SEGA esta totalmente probado y funcional.

Sin embargo, para simular las comunicaciones con SEGA, sí que se dispone de ficheros de comunicación iguales a los que utiliza y que son copias de otros ficheros que anteriormente han funcionado correctamente, por tanto, estos ficheros se utilizarán para probar las comunicaciones con SEGA

4.2.2. Pruebas unitarias

Las pruebas unitarias servirán para comprobar si el funcionamiento de algunos métodos en concreto, que tienen una mayor extensión o complicación, es correcto. No se van a probar todos

los métodos del sistema ya que algunos de ellos tienen un funcionamiento muy trivial, y otros dependen tanto de las comunicaciones con otros sistemas que su correcto funcionamiento ya se comprobará explícitamente en las pruebas de integración.

- Lector de ficheros. Estos métodos son los encargados de descomponer la información de los ficheros de comunicación procedentes de SEGA, y dejar su información en una *DataTable* que luego devuelven. Para probar su funcionamiento, se dispone de dos ficheros de cada tipo, uno que se sabe que esta formado correctamente y contiene información verídica, y otro que tiene problemas concretos en su estructura.

  Además de esos ficheros también se dispone de la DataTable exacta que debe generar cada fichero. Por tanto, para probar su correcto funcionamiento se harán las siguientes pruebas unitarias:

  - Para cada tipo de mensaje se realiza un test diferente en el que se le pasa como argumento el fichero correspondiente que contiene la información correcta. Tras lo cual se comprueba que la tabla devuelta por el método corresponde exactamente con la tabla que se sabe que es correcta.
  - Para cada tipo de mensaje se realiza un test diferente en el que se le pasa como argumento el fichero correspondiente que contiene información mal formada. Se espera que este método falle y devuelva un valor nulo. Por lo cual sabemos que no se ha podido leer el fichero correctamente.

  Como dato cabe indicar que no se comprueba que la información en sí sea correcta, sino solamente que este bien formado el fichero según el protocolo de comunicación. La responsabilidad de comprobar si la información es correcta corresponde a cada tarea.

- Lector de la configuración de las tareas. Este método es el encargado de leer el fichero xml en el que se encuentra la configuración de las tareas, y arrancar los servicios en función de esa configuración.

  Para probar el funcionamiento de este método se dispone de tres ficheros xml, uno de ellos contiene información de configuracion de 7 tareas las cuales ya estan programadas y funcionan correctamente. Por otro lado, el otro fichero que sí esta formado correctamente como fichero xml, pero contiene información que no es correcta, como un identificador que no corresponde a ninguna tarea y tiempos negativos. Y finalmente otro fichero de configuracion que tiene una estructura errónea de fichero xml.

  - Para la prueba con el fichero correcto, se le pasa al método como argumento para que lo lea, tras lo cual, se comprueba que el número de tareas sea el correcto, que cada tarea instanciada sea la correcta y que sus tiempos de ejecución sean los adecuados. Esta información que se sabe que es la correcta se carga previamente en una lista.
  - Para la prueba con el fichero bien formado pero con información errónea, se comprueba que el método no instancia ninguna tarea y lanza una excepción del tipo ArgumentException. Si al realizarse la prueba se obtiene otra excepción o un valor nulo, el método se considera erróneo.
  - Para la prueba con el fichero mal formado, es decir que no tiene el número de atributos esperados, se comprueba que el método nativo que lee el xml detecta que faltan atributos al leer usando un esquema xml como referencia. Al encontrar ese error, el método debería lanzar una excepción del tipo InvalidOperationException.

- Metodo comprobar si la tarea esta disponible. Este método es el encargado de, una vez el timer que tiene asignado cada tarea termine, comprobar el estado de la tarea y, en caso de que este disponible ejecutarla. También debe comprobar que si es una tarea que se tiene que ejecutar a una hora del día en concreto, sea la hora correcta

  Para probar el funcionamiento de esta tarea se dispone de cinco tareas distintas ya instanciadas. Una de ellas esta totalmente disponible y sin errores,la segunda esta disponible y tiene asignada una hora del día, la cual es la hora actual, la otra se simula que todavía esta ejecutando la iteración anterior, la cuarta está disponible pero generó errores graves de ejecución y la última esta disponible pero no es la hora del día que corresponde.

  - Para comprobar la tarea disponible, simplemente se la damos al método como argumento y esperamos que nos devuelva un booleano verdadero.
  - Para comprobar la tarea que se ejecuta a una hora del día que es ahora, primero se asigna a la tarea la hora actual del sistema, luego se manda al método como argumento y se espera una respuesta booleana positiva.
  - Para comprobar la tarea que sigue ocupada con la tarea anterior, simplemente se la pasamos al método como argumento y esperamos un booleano falso.
  - Para comprobar la tarea disponible con errores graves, le pasamos al método la tarea como argumento. Al se ser un error grave, la tarea no debería ejecutarse y por tanto, debería devolver una un booleano nulo.
  - Para comprobar la tarea que no esta en la hora del día correspondiente, primero se cambia la hora del ordenador a una que no corresponda a la tarea, luego se pasa esa tarea al método como argumento y se espera un booleano false.

- Creador de ficheros. Estos métodos son los encargados de, una vez que se dispone de la información que se tiene que depositar en los ficheros de comunicación en una DataTable, crear un fichero nuevo y volcar ahí la información.

  Para probar este funcionamiento se dispone de tres ficheros de comunicación que se sabe que son correctos, uno por cada tarea que tiene que escribir mensajes de comunicación. Además también se tienen las DataTable que generan cada fichero en concreto. Además de eso también se dispone de DataTable que tienen información con mala estructura, es decir, que no tienen todos los campos que deberían.

  - Para cada tipo de fichero se realiza un test diferente en el que se pasa como argumento la DataTable. Tras lo cual se comprueba si el fichero que ha generado el método se corresponde exactamente con el fichero que se sabe que es correcto. Si son iguales la prueba se considera correcta.
  - Para comprobar las DataTable mal formadas, estas se pasan como argumento al método y se espera como resultado que lancen una excepción de ArgumentException y que además no generen ningún tipo de fichero. Si esto ocurre se considera que la prueba es correcta.

  Como dato comentar que no se comprueba que la información en si sea correcta, sino solamente que este bien formada la DataTable que contiene la información. La responsabilidad de comprobar si la información es correcta corresponde a la tarea que llama al método.

4.2.3. Pruebas integración

En las pruebas de integración se va a probar si las comunicaciones del sistema con los demás sistemas y todos los procedimientos necesarios para llegar a esa comunicación dan como lugar el resultado deseado. Lo ideal sería poder probar las comunicaciones tanto del sistema con AX como con SEGA, sin embargo, como SEGA no tiene entorno de pruebas y su comunicación termina en el momento que se deposita un fichero en su directorio, no es posible probar nada de integración con sus sistema. En cambio, para la integración con AX si que se puede utilizar su base de datos de desarrollo y comprobar los cambios reales que experimentaría el sistema.

- Cambio de inventario. Esta tarea en la encargada de, una vez recibido un mensaje de SEGA donde estan las últimas transacciones de inventario, avisar a AX para que ajuste su stock, es decir, que el resumen de las cantidades de los productos aumente o disminuya según corresponda. Para probar correctamente esta funcionalidad se suponen los siguientes escenarios:

  - Fichero correcto. Se dispone de un fichero que se sabe seguro que esta bien formado y que tiene el objetivo de aumentar y disminuir varias veces el stock de dos productos concretos. Para probar el correcto funcionamiento de la tarea, se inicia el servicio y se deja ese fichero para su recolección por la tarea. Primero se comprueba el stock actual de esos productos, como la base de datos es de un entorno de prueba que nadie más utiliza se sabe que esos stock no van a cambiar durante la ejecución. Cuando termina la ejecución de la tarea, se comprueba que el stock final es el correcto según el fichero mandado.
  - Fichero casi correcto. Se dispone un fichero con muchas líneas en la que una de ellas tiene un problema grave en su estructura. Como la ejecución de todas las líneas tiene que ser atómica, hay que comprobar que al final no se ha contabilizado ninguna línea y por tanto el stock de los productos del mensaje no ha cambiad. Para comprobar esto se mira el stock ante de empezar, luego se deposita el mensaje, y cuando la tarea termina su ejecución se vuelve a comprobar el stock.
  - Fichero correcto pero comunicación caída. Como el sistema debe responder correctamente a las caídas de comunicaciones, se realiza una prueba en la que el servicio web no esta disponible. Para simular este funcionamiento se va a utilizar una clase *mock* o "de prueba"que sustituye al servicio web. Para la ejecución del test se inyecta primero la clase *mock* y luego se deposita un fichero de cambio de inventario correcto. La clase *mock* esta configurada para que cuando se llame al método de comunicación del servicio web, este conteste con un error de conexión. Tras la ejecución de la tarea se comprueba que el sistema lanza la excepción correcta y se registra la incidencia correctamente en el log.

- Cerrado ruta de picking. Esta es la tarea encargada de, una vez que se ha completado una ruta de picking, leer el mensaje del gestor del almacén que lo indica, y con la información del mensaje avisar a AX para que cierre la ruta de picking y prepare el albarán de envío.

  - Fichero correcto cierra ruta existente. Se dispone de un fichero que se sabe que es correcto y que cierra la ruta de picking de un envío que se sabe que existe en el entorno de pruebas. Para realizar la prueba primero se comprueba que el estado de

la ruta de picking del envío es correcto. Tras lo cual se deposita el fichero que cierra esta ruta. Finalmente una vez termina la ejecución se comprueba que el estado de la ruta de picking es cerrado y que la del envío es a la espera. Además también se comprueba que el número de albarán generado es correcto.

- Fichero correcto pero ruta no existente. Se dispone de un fichero que se sabe que es correcto y que cierra la ruta de picking de un envío que se sabe que no existe en el entorno de pruebas. Para realizar la prueba primero se comprueba que de verdad esa ruta de picking no existe. Luego se deposita el fichero y se espera que el sistema lance la excepción esperada. Finalmente se comprueba que no se ha creado ninguna ruta de picking nueva y que la incidencia se ha registrado correctamente en el sistema.
- Fichero incorrecto. Se dispone de un fichero que se sabe que no es correcto y que no cierra ninguna ruta de picking. Para realizar la prueba se deposita el fichero con errores y se espera a que el sistema lance una excepción de fichero incorrecto. Tras lo cual también se comprueba que la incidencia se ha registrado correctamente.

Capítulo 5

Conclusiones

En este capítulo se van a relatar las diferentes conclusiones a las que se ha llegado con el desarrollo de este proyecto. Estas conclusiones se van a agrupar en tres ámbitos diferentes. Primero las conclusiones en el ámbito formativo sobre lo que he aprendido, luego en el ámbito profesional y mi experiencia en la empresa, y finalmente, sobre el ámbito personal.

- En el ámbito formativo, el desarrollo de este proyecto me ha servido en muy buen grado para ampliar mis conocimientos de programación, principalmente en el ámbito del desarrollo en backend y en el desarrollo de aplicaciones para escritorio con .NET. Otra cosa a destacar de lo aprendido es el manejo de bases de datos grandes y complejas, en este sentido he aprendido cómo se organizan las bases de datos dentro de una empresa relativamente grande y que tiene que gestionar mucha información, sobre todo en como hace esto último un programa ERP. Y finalmente destacar todo lo aprendido en referencia a la necesidad de utilizar un buen ERP en una empresa para manejar la información y en como estos tipos de programas funcionan.
- En cuanto al ámbito profesional y mi experiencia en la empresa, me ha servido para comprender el trabajo de un ingeniero informático en una empresa que no se dedica a la informática, sino que esta sirve de apoyo para las demás operaciones. También he aprendido ha trabajar en equipo dentro del mismo departamento y cómo realizar un proyecto que depende del trabajo de otras personas para desarrollarse. Finalmente, destacar la utilidad e importancia de mantener unas buenas comunicaciones entre compañeros para evitar malentendidos y trabajo repetido.
- En cuanto al ámbito personal, este proyecto me ha servido para darme cuenta que dentro del ámbito de la informática todavía me quedan muchas cosas que aprender. Del mismo modo, también me he dado cuenta que todas esas cosas que me faltan por aprender se pueden lograr con un poco de esfuerzo y ganas. También quiero añadir que me hubiera gustado poder acabar completamente el desarrollo del sistema y poder implementarlo, pero por motivos de la empresa no ha sido posible.

Capítulo 6

Bibliografía

[1] Gustav Karner. (1993) metrics for objectory. university of linköping, sweden.

[2] nlog-project. (05-06-2019) nlog para c#, biblioteca que gestiona el sistema de log y sus ficheros. `https://nlog-project.org/`.

Anexo A

Documento Protocolo de comunicación SEGA-MOVEX

tecsidel

TECNOLOGÍAS DE LA INFORMACIÓN

Manual de Interfaces SEGA - MOVEX
Marie Claire S.A.

Edición: **1.0.13**

Autor: **Arseni Lago**

Fecha: **6 de Mayo 2005**

Historia del Documento

Versión: **1.0.0** *Descripción*: Versión inicial

Elaborado por: **Roberto Rodríguez** *Fecha*: **15/04/2004**

Revisado por: *Fecha*:

Aprobado por: *Fecha*:

Versión: **1.0.1** *Descripción*: Actualización del documento después de reuniones:

- 20/04/2004 Interfaces ERP - SEGA
- 22/04/2204 Interfaces SCT – SEGA

Elaborado por: **Arseni Lago** *Fecha*: **21/04/2004**

Revisado por: **Roberto Rodríguez** *Fecha*: **26/04/2004**

Aprobado por: *Fecha*:

Versión: **1.0.2** *Descripción*:

- Modificación del formato de los nombres de los ficheros de interface
- Corrección del fichero CS

Elaborado por: **Arseni Lago** *Fecha*: **26/04/2004**

Revisado por: *Fecha*:

Aprobado por: *Fecha*:

Versión: **1.0.3** *Descripción*: Actualización del documento después de reunión 28/04/2004

Elaborado por: **Arseni Lago** *Fecha*: **29/04/2004**

Revisado por: **Roberto Rodríguez** *Fecha*: **30/04/2004**

Aprobado por: *Fecha*:

Versión: **1.0.4** *Descripción*:

- Inclusión de Provincia Entrega en fichero PS

Elaborado por: **Arseni Lago** *Fecha*: **05/05/2004**

Revisado por: *Fecha*:

Aprobado por: *Fecha*:

Versión: **1.0.5** *Descripción*:

- Inclusión de motivos de ajuste en fichero AS

Elaborado por: **Arseni Lago** *Fecha*: **06/05/2004**

Revisado por: *Fecha*:

Aprobado por: *Fecha*:

Versión: **1.0.6** *Descripción*:

- Precisiones sobre la generación de los ficheros CE

Elaborado por: **Arseni Lago** *Fecha*: **26/05/2004**

Revisado por: *Fecha*:

Aprobado por: *Fecha*:

Versión: **1.0.7** *Descripción*:

- Inclusión del campo ListaContenido en PS

Elaborado por: **Arseni Lago** *Fecha*: **4/06/2004**

Revisado por: *Fecha*:

Aprobado por: *Fecha*:

Versión: **1.0.8** *Descripción*:

- Inclusión del identificador de línea Movex en CP

Elaborado por: **Arseni Lago** *Fecha*: **1/07/2004**

Revisado por: *Fecha*:

Aprobado por: *Fecha*:

Versión: **1.0.9** *Descripción*:

- Cambio de las cantidades (packs, unidades) en AE, PS, CE, CP, AS, SA

Elaborado por: **Arseni Lago** *Fecha*: **12/07/2004**

Revisado por: *Fecha*:

Aprobado por: *Fecha*:

Versión: **1.0.10** *Descripción*:

- Cambio en MA para pasar las Cajas por Capa de N4 a N5

Elaborado por: **Arseni Lago** *Fecha*: **29/11/2004**

Revisado por: *Fecha*:

Aprobado por: *Fecha*:

Versión: **1.0.11** *Descripción*:

- Inclusión de un nuevo valor en el Localización del MA para los artículos del tipo 'Picking Sin Sorter'

Elaborado por: **Arseni Lago** *Fecha*: **15/04/2005**

Revisado por: *Fecha*:

Aprobado por: *Fecha*:

Versión: **1.0.12** *Descripción*:

- Inclusión de la descripción de los ficheros CP en el caso de pedidos anulados
- Actualización de las condiciones para el envío de los ficheros CS

Elaborado por: **Arseni Lago** *Fecha*: **22/04/2005**

Revisado por: *Fecha*:

Aprobado por: *Fecha*:

Versión: **1.0.13** *Descripción*:

- Inclusión de nuevos tipos de manipulación en los pedidos (PS)

Elaborado por: **Arseni Lago** *Fecha*: **06/05/2005**

Revisado por: *Fecha*:

Aprobado por: *Fecha*:

Índice:

1 Introducción

Este documento contiene la descripción del procedimiento de comunicación y del contenido de la información intercambiada entre el sistema de gestión del almacén (SEGA) y los sistemas de gestión comercial (Movex).

Movex envía a SEGA información sobre los artículos, familias, órdenes de compra, y pedidos; mientras que SEGA envía a Movex información de la situación y movimientos de stock del almacén (confirmación de entradas, confirmación de salidas, movimientos de stock, stock actual, etc).

2 Comunicación entre Movex y SEGA

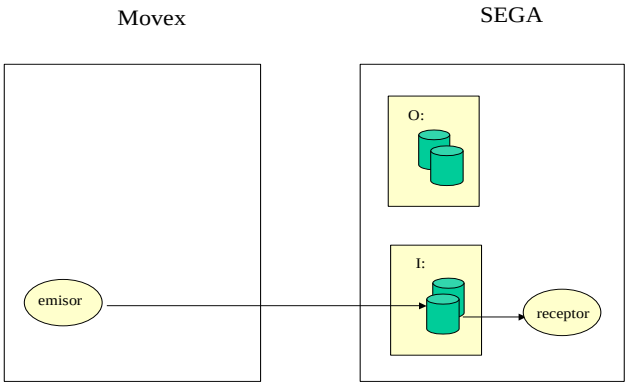
La comunicación entre SEGA y Movex se efectuará mediante ficheros depositados en dos directorios (uno de entrada y otro de salida) del sistema SEGA, a los que tiene acceso Movex.

En este documento se utilizará de forma indistinta el término fichero o mensaje para referirse a la información intercambiada entre los dos sistemas.

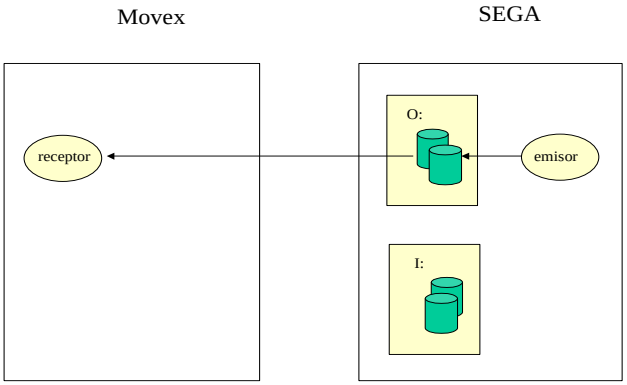
SEGA dispone de dos directorios locales:

- I: (entrada) para la recepción en SEGA de los ficheros enviados desde Movex
- O: (salida) para el envío de ficheros desde SEGA a Movex.

Para el envío de datos de Movex a SEGA, Movex deposita en el directorio I: de SEGA uno o varios ficheros con los datos a enviar. El tratamiento de los ficheros enviados por Movex se hará de forma automática. Después de importar un fichero, SEGA lo marca como tratado (lo mueve al directorio I:\Backup).



Para el envío de datos de SEGA a Movex, SEGA crea el fichero en el directorio O: y espera que Movex lo recupere.



Es responsabilidad de Movex la detección de los ficheros preparados por SEGA y pendientes de ser recibidos, así como el procedimiento para distinguir los que ya están tratados, de los que todavía están pendientes de tratamiento (transferencia del fichero a otro directorio O:\backup, o cambio de la extensión del fichero, o eliminación del fichero).

2.1 Convenciones de descripción de formatos

Para describir el contenido de los ficheros intercambiados, se utiliza la siguiente convención:

- **Cn**: es un campo alfanumérico de n caracteres. El relleno será con espacios por la derecha.
- **Nx.y**: es un campo de longitud x caracteres, todos numéricos, justificados a ‘0’ por la izquierda, considerando que los y caracteres de la derecha son decimales (el punto decimal no se escribe). Puede incluir un signo (en la primera posición de la izquierda).
- **Nx**: se considera equivalente a Nx.0
- **B**: es un campo de tipo C1 en el que los únicos valores admisibles son ‘S’ y ‘N’ (booleano) (‘S’ = True; ‘N’ = False).
- Los campos opcionales se informan con espacios si son alfabéticos, o con 0 si son numéricos.

2.2 Nomenclatura de los ficheros de interface

Para garantizar que el tratamiento de los ficheros de comunicación se efectúa en el orden requerido, se utilizan las siguientes reglas de nomenclatura de los ficheros de comunicación:

- **Tipo de sistema origen** (C3): MVX, SGA. Movex prefija sus ficheros mediante MVX y SEGA prefija los suyos con SGA. Estos prefijos son los sugeridos, aunque son completamente parametrizables.
- **Identificador del sistema origen** (N4): cuando exista más de un sistema origen del mismo tipo. En el caso normal de existir sólo uno, se pondrá 0001.
- **Tipo de fichero** (C2): identifica el tipo de contenido del fichero (tipo de mensaje, por ejemplo: PS, CE, MA, etc.).
- **Fecha y hora** (N12): fecha y hora de generación, con el formato AAAAMMDDHHMM.
- **Número de secuencia** (N5): número secuencial para poder distinguir los ficheros del mismo tipo que se envíen en el mismo minuto. Puede ser un correlativo que se reinicia al alcanzar el valor 100000. Actualmente los ficheros enviados desde Movex a SEGA tienen un correlativo de N4. En los ficheros enviados de SEGA a Movex el correlativo se interpreta de la forma siguiente: SS###, donde SS son los segundos de la fecha y hora de generación, y ### es el número de orden de un fichero dentro del segundo en que se ha generado (empezando por 001).
- **Extensión**: .TXT

Por ejemplo, nombres de ficheros de interface pueden ser:

MVX0001MA20040421135200001.TXT (fichero de maestro de artículos, enviado desde Movex)

SGA0001CE20040421102317001.TXT (fichero de confirmación de entradas, enviado a Movex)

2.3 Estructura de los ficheros de interface

En el interior de un fichero de comunicación, existirán unos indicadores (tags) del inicio y fin del fichero, así como del nivel de la información (útil para describir información jerarquizada: maestro, detalle de 1er. nivel, detalle de 2º nivel, ...):

[INICIO]<intro> : será la primera línea del fichero.

[H1]<intro> : indicará que se inicia un bloque de registros a nivel 1 (sería el correspondiente a una cabecera)

[H2]<intro> : indicará que se inicia un bloque de registros a nivel 2 (sería el correspondiente a las líneas de detalle del nivel 1. Se admite cualquier nivel de detalle (H3, H4, …)

[FIN]<intro> : será la última línea del fichero.

Dentro de un nivel, los registros se separan por <intro> (retorno de carro). Dentro de un registro los campos (todos de longitud fija), se colocan seguidos, sin separadores.

2.4 Protocolo de comunicación

La forma de actualización de los maestros puede ser on-line o de forma batch. Cuando SEGA trate un conjunto de ficheros recibidos, lo hará teniendo en cuenta la fecha de generación del fichero.

Los ficheros se procesan por orden de fecha de creación.

Movex deberá generar los ficheros en el orden adecuado para su tratamiento:

Tabla	Depende de
MA	MF
AE	MA
PS	MA

- La importación de artículos requiere que las familias correspondientes estén importadas
- La importación de avisos de entrega y de pedidos requiere que los correspondientes artículos estén importados

Para cada fichero recibido e importado por SEGA, éste enviará a Movex una confirmación de recepción (ver el parágrafo “Confirmación de mensaje (‘CF’)”), para poder asegurar que no se pierden mensajes.

En SEGA se mantendrá en los ficheros de log una traza de todos los ficheros enviados y recibidos.

Si SEGA descarta una línea de detalle de un aviso de entrada o de una entrega a preparar, entonces se descarta el aviso de entrada o la entrega a preparar completa.

En el caso de importación de ficheros maestros, sólo se importan los registros correctos.

Movex debe controlar que los ficheros enviados por SEGA son procesados correctamente.

2.4.1 Fichero de resultado de la importación

Para cada fichero importado en SEGA, además del mensaje de confirmación, se generará un fichero con el nombre del fichero original recibido por SEGA, con extensión OUT, con información del resultado de la importación.

Por ejemplo, si se ha importado un aviso de entrega MVX0001AE999999999999999.txt, se generará un fichero SGA0001AE999999999999999.OUT, con la información siguiente:

FICHERO: I:\MVX0001AE999999999999999.txt

FECHA: 2001/02/21 16:32:40

Orden de compra [0000000000000000999]: linea descartada, motivo [El articulo [0000000978132] es de crossdocking y no tiene reparto, LA LINEA SE DESCARTA], registro [00000009781320000132S]

Orden de compra [0000000000000000999]: linea descartada, motivo [El articulo [0000000919739] es de crossdocking y no tiene reparto, LA LINEA SE DESCARTA], registro [00000009197390000133S]

Insertados: 1

Modificados: 0

Rechazados: 0

Total: 1

En la primera línea se incluye el nombre del fichero.

En la segunda línea se incluye la fecha de importación en SEGA.

Entre las líneas de trazos se incluyen los posibles errores de importación.

En las siguientes líneas se incluye un resumen con la cantidad de registros insertados, modificados, rechazados, y el número total.

3 Descripción de los ficheros de interface

Movex → SEGA	SEGA → Movex
Maestro de familias (MF)	Confirmación de entradas al almacén (CE)
Maestro de artículos (MA)	Confirmación de preparación (PS)
Maestro de centros (MX)	Confirmación de salidas del almacén (CS)
Aviso de entrega (AE)	Ajustes de stock (AS)
Entregas a preparar (PS)	Stock actual de artículos (SA)
	Confirmación mensaje (CF)

3.1 Ficheros generados por Movex y enviados a SEGA

3.1.1 Maestro de familias (‘MF’)

Contiene las altas o modificaciones del maestro de familias.

- Evento que dispara la transmisión a SEGA: el alta o la modificación en el maestro de familias de algún dato necesario en SEGA
- Proceso que es disparado en SEGA por la transmisión: actualización del maestro de familias.

Información transmitida

Nivel	Campo	Descripción	Tipo	Obligatorio
[H1]				
	Identificador Familia	Código de la familia (debe ser único)	C8	Si
	Descripción		C35	Si

3.1.2 Maestro de artículos ('MA')

Contiene las altas, modificaciones o bajas lógicas del maestro de artículos.

- Evento que dispara la transmisión a SEGA: el alta o la modificación de algún dato del maestro de artículos que afecte al almacén
- Proceso que es disparado en SEGA por la transmisión: actualización del maestro de artículos.

La terminología empleada en este documento para las diferentes agrupaciones de productos es la siguiente:

SEGA	Marie Claire	Descripción
Unidad	**Envase**	Unidad de consumo por parte del cliente final
Pack	**Caja**	Unidad mínima de venta de Marie Claire, excepto para los pedidos de tipo NSD. Contiene n envases
Caja	**Embalaje**	Unidad de trasporte que contiene n cajas

Información transmitida

Nivel	Campo	Descripción	Tipo	Obligatorio
[H1]				
	Código SKU	Código interno de Marie Claire (debe ser único)	C15	Si
	EAN unidad	EAN13 de la unidad (en blanco si no existe)	N13	Si (**)
	EAN pack	EAN13 del pack (debe ser único)	N13	Si
	EAN caja	DUN14 del embalaje (en blanco si no existe)	N14	
	Identificador Familia	Código de la familia	C8	Si
	Rotación	Rotación: '01': AAA (rotación A de Movex) '02': AA (rotación B de Movex) '03': A (rotación C de Movex) '04': B (rotación D de Movex) '05': C (rotación E de Movex) '06': D (rotación F de Movex) '07': E (rotación G de Movex) '08': F (rotación H de Movex) '09': G (rotación I de Movex) '10': H (rotación J de Movex) '11': I (rotación M de Movex)	N2	Si

	Descripción Corta	Descripción corta, se utiliza en pantallas de RF Deberá contener la información necesaria para determinar el artículo por un operario de RF	C20	Si
	Descripción Larga	Descripción larga, se utiliza en la documentación impresa y en ventanas de la aplicación cliente. Descripción (C30) + Color/Talla (C30)	C60	Si
	Alto Unidad	Corresponde a la altura, en milímetros, de una unidad de producto. Valores de 00001 a 99999	N5	Si (**)
	Ancho Unidad	Igual que la anterior, pero para la dimensión de anchura	N5	Si (**)
	Largo Unidad	Igual que la anterior, pero para la dimensión de longitud	N5	Si (**)
	Peso Unidad	Peso en gramos de una unidad de producto, de 00000001 a 99999999	N8	Si (**)
	Factor Compactación Unidad	Factor de compactación de la unidad (por defecto 1000)	N4.3	Si (**)
	Alto Pack	Corresponde a la altura, en milímetros, de un pack de producto. Valores de 00001 a 99999	N5	Si
	Ancho Pack	Igual que la anterior, pero para la dimensión de anchura	N5	Si
	Largo Pack	Igual que la anterior, pero para la dimensión de longitud	N5	Si
	Peso Pack	Peso en gramos de un pack de producto, de 00000001 a 99999999	N8	Si
	Factor Compactación Pack	Factor de compactación del pack (por defecto 1000)	N4.3	Si
	Unidades Por Pack	Número de unidades en un pack. Valores de 00001 a 99999. Si no tiene unidades, se pone 1 (pack = unidad)	N5	Si
	Unidades Por Caja	Número de unidades en una caja. Valores de 00001 a 99999. Si no existe la caja, este valor se informará a unidades por pack (caja = pack)	N5	Si
	Cajas Por Capa	Número de cajas en una capa o nivel de palet. Valores de 00001 a 99999. Si no existe la caja se pone el número de packs por capa	N5	Si
	Capas Por Palet	Número de capas o niveles en un palet completo. Valores de 0001 a 9999.	N4	Si
	Localización	Tipo de localización del artículo (a definir). Ej: ‘10’: Picking ‘20’: Trilateral ‘30’: Estación de promocionales ‘40’: Picking de promocionales ‘50’: Picking sin sorter	N2	Si
	Baja Lógica	Indicador de que el artículo no se puede recibir, preparar, expedir.	B	Si (*)

(*) Es obligatorio para dar la baja lógica de un SKU.

(**) Son obligatorios para aquellos SKU que tengan unidades.

3.1.3 Maestro de centros (‘MX’)

Contiene las altas o modificaciones del maestro de centros. En SEGA se denomina centro a cualquiera de los diferentes clientes, proveedores, transportistas o almacenes con los que el almacén actual puede intercambiar (recibir o expedir) mercancía. Un centro se identifica por dos valores: el tipo de centro (cliente, proveedor, transportista, otros almacenes, etc.), y su código de centro.

No es necesario que Movex transfiera a SEGA todo el maestro de centros. De momento parece recomendable transferir los centros de tipo transportistas, los centros de tipo proveedores y los otros almacenes (fábrica de Villafranca, devoluciones, etc.).

- Evento que dispara la transmisión a SEGA: el alta o la modificación de algún dato del maestro de clientes, proveedores, transportistas, u otros almacenes, que afecte al SEGA
- Proceso que es disparado en SEGA por la transmisión: actualización del maestro de centros.

Información transmitida

Nivel	Campo	Descripción	Tipo	Obligatorio
[H1]				
	Código Tipo Centro	Tipo de centro. Valores por defecto: ‘1’: cliente ‘2’: proveedor ‘3’: transportista ‘4’: otros almacenes Estos valores pueden ser modificados, mediante la actualización de la tabla TIPOS_CENTRO del SEGA	C15	Si
	Código Centro	Identificador del centro, no puede haber dos códigos iguales dentro del mismo tipo de centro (*)	C15	Si
	Nombre	Nombre o razón social	C40	Si

(*) Código Centro:

Si es un proveedor: Código Proveedor (N10) + 5 blancos

Si es un almacén: Código Almacén (N3) + 12 blancos

3.1.4 Aviso de entrega de orden de entrada (‘AE’)

Es el aviso de que nueva mercadería será recibida por el almacén. Este fichero equivale a la orden de compra o de entrada. La transmisión desde Movex será on-line.

- Evento que dispara la transmisión a SEGA:
  - Orden de distribución desde los almacenes de producción de Castellón o Villafranca con destino al Centro de Distribución de Borriol. Incluye devoluciones y manipulación
  - La confirmación de envío de una orden de compra a terceros
- Proceso que es disparado en SEGA por la transmisión: inclusión de la nueva orden de compra en la tabla de entradas previstas.

Restricciones de tratamiento:

- Si no se ha empezado a recibir, puede importarse de nuevo (se elimina la anterior).
- Si alguno de los registros es incorrecto, se rechaza toda la orden de entrada.

Información transmitida

Nivel	Campo	Descripción	Tipo	Obligatorio
[H1]				
	Identificador Orden Entrada	Identificador único de la orden de entrada. Este identificador no debe repetirse nunca (*)	N18	Si
	Tipo Orden Entrada	Tipo de la orden de entrada: ‘01’: orden de compra de terceros ‘02’: no aplicable ‘03’: transferencia de otro almacén	N2	Si
	Código Centro Origen	Código del proveedor o almacén	C15	Si
	Fecha Entrega	Fecha de entrega: AAAAMMDD	N8	
[H2]				
	Posición	Número de línea de la orden de entrada en Movex (**)	C17	Si
	Código SKU	Código del artículo	C15	Si
	Cantidad	Cantidad esperada en packs	N9	Si

(*): Identificador Orden Entrada: Identificador de Tipo de Entrega en Movex + 000000 + Identificador de Entrega

Correspondencia entre tipos de entrega Movex y SEGA:

Movex	SEGA	Descripción
25	**01**	Orden de compra de terceros
50	**03**	Transferencia de otro almacén

(**): Número de línea de la orden de entrada en Movex: número de la orden de entrada (11) + número de línea (6)

3.1.5 Entregas a preparar ('PS')

- Evento que dispara la transmisión a SEGA:
  - Cuando la entrega a cliente pasa al estado de picking
  - Cuando se genera una orden de fabricación se surtidos
- Proceso que es disparado en SEGA por la transmisión: Almacenamiento de las entregas a preparar que posteriormente entran en el proceso de selección de entregas a preparar por el CD al día siguiente.

Restricciones de tratamiento:

- Si no se ha empezado a procesar, puede importarse de nuevo (se elimina la anterior).
- Si alguno de los registros es incorrecto, se rechaza toda la entrega.

Información transmitida

Nivel	Campo	Descripción	Tipo	Obligatorio
[H1]				
	Número Entrega	Identificador único de la entrega (se corresponde con el albarán y equivale a un pedido SEGA) (*)	N15	Si
	Tipo Entrega	Tipo de la entrega: '01': normal '02': NSD '03': NPI. Aparece en las L15 de envio a fábrica. Sale al sorter. '04': NPI. Aparece en las L12 que lanzamos. Cuando se lanza, se poner automáticamente en picking Express.	N2	Si
	Tipo Manipulación	Tipo de manipulación: '01': sin manipulación '02': manipulación sin cambio de contenido de cajas '03': Manipulación con cambio de contenido de cajas (en cubetas) '04': sin manipulación, requiriendo documentación especial de exportación '05': manipulación sin cambio de contenido de cajas, requiriendo documentación especial de exportación '06': Manipulación con cambio de contenido de cajas (en cubetas) , requiriendo documentación especial de exportación '07': Picking surtidos. Od con destino MA1 para el montaje de surtidos.	N2	Si
	Urgencia	Urgencia de preparación del pedido (01 a 99). Cuanto más pequeño es más urgente. El valor correspondiente a urgente es el 01.	N2	Si
	Fecha Inicio Preparación	Fecha prevista de inicio de preparación: AAAAMMDD	N8	Si
	Nombre Cliente	Nombre del cliente	C40	
	Representante	Código del representante + Nombre del representante	C40	
	Dirección Fiscal 1	Primera línea de la dirección fiscal	C40	
	Dirección Fiscal 2	Segunda línea de la dirección fiscal	C40	
	Dirección Fiscal 3	Tercera línea de la dirección fiscal	C40	
	Dirección Fiscal 4	Cuarta línea de la dirección fiscal	C40	
	Dirección Entrega 1	Primera línea de la dirección de entrega	C40	

	Dirección Entrega 2	Segunda línea de la dirección de entrega	C40	
	Dirección Entrega 3	Tercera línea de la dirección de entrega	C40	
	Dirección Entrega 4	Cuarta línea de la dirección de entrega	C40	
	Provincia Entrega	Provincia de entrega	C30	
	Código Postal Entrega	Código postal de entrega	C5	
	Observaciones Etiqueta	Observaciones que aparecen en la etiqueta de envío y en la hoja de ruta	C40	
	Observaciones Albarán	Observaciones que aparecen en el albarán (al final del pedido)	C300	
	Observaciones Manipulación	Observaciones que aparecen en la pantalla de manipulación	C300	
	Su proveedor	Código del proveedor (Marie Claire) desde el punto de vista del cliente	C15	
	Código Agencia	Código de la agencia de transporte (10 dígitos + 5 blancos)	C15	Si
	Tipo Etiqueta	Tipo de etiqueta de envío a generar: '01': etiqueta tipo Marie Claire '02': etiqueta tipo Azkar '03': etiqueta tipo Gómez	N2	Si
	Código Barras	Código de barras del bulto para la agencia	C19	Si
	Código Barras Legible	Código de barras legible del bulto para la agencia	C22	Si
	Albarán Valorado	Indicador de si se debe imprimir un albarán valorado	B	Si
	Código Tipo Caja	Código del tipo de caja a utilizar en los pedidos Identificador de Tipo de cajas en Movex + 5 blancos Si no viene informado, se utiliza el tipo más grande de los generales	C15	
	Lista Contenido	Indicador de si hay que imprimir lista de contenido para cada caja del pedido (S, N)	B	Si
[H2]				
	Pedido Movex	Identificador del pedido en Movex	C10	Si
	Posición	Número de línea del pedido en Movex: pedido Movex (N10) + Línea Movex (N6)	C16	Si
	Pedido Cliente	Identificador del pedido del cliente	C35	
	Su Sección-Departamento	Sección y departamento para el cliente	C30	
	Sección	Sección a la que pertenece el artículo	C35	
	Marca	Marca a la que pertenece el artículo	C35	
	Código Artículo Marie Claire	Código de artículo para Marie Claire	C5	

	Código Artículo Cliente	Código de artículo para el cliente	C13	
	Código SKU	Código del artículo	C15	Si
	Cantidad	Cantidad a servir (en packs si el tipo de pedido es normal, en unidades si el tipo de pedido es NSD)	N9	Si
	Precio Venta	Precio de venta	N6.3	Si

(*): Número Entrega: Tipo de Entrega según Movex + 00 + Identificador de Entrega a preparar

3.2 Ficheros generados por SEGA y enviados a Movex

3.2.1 Confirmación de entradas ('CE')

Es la confirmación a Movex de cualquier entrada de mercadería que se haya producido en el almacén, ya sea de un proveedor, o de una transferencia de otro almacén.

Si en una recepción vienen varias órdenes de entrada, se genera un fichero para cada orden de entrada recibida.

- Evento que dispara la transmisión de SEGA: la finalización de una de las recepciones en SEGA
- Proceso que es disparado en Movex por la transmisión: actualización de stock en el almacén, actualización contable, ...

Información transmitida

Nivel	Campo	Descripción	Tipo	Obligatorio
[H1]				
	Identificador Orden Entrada	Identificador de la orden de entrada	N18	Si
	Fecha	Fecha de la recepción: AAAAMMDD	N8	Si
	Hora	Hora de la recepción: HHMMSS	N6	Si
	Número Albarán	Albarán emitido por el almacén o proveedor origen	C18	
	Número Entrega SEGA	Correlativo único generado por SEGA para cada una de órdenes de entrada recibidas en la recepción	N18	Si
[H2]				
	Posición	Número de línea de la orden de entrada en Movex	C17	Si
	Código SKU	Código del artículo	C15	Si
	Cantidad	Cantidad recepcionada en packs	N9	Si

3.2.2 Confirmación de entrega a cliente (‘CP’)

Es la confirmación a Movex del fin de preparación de una entrega a un cliente, indicando la lista de los contenedores preparados y su contenido.

- Evento que dispara la transmisión de SEGA: la finalización de la preparación de un pedido, antes de ser expedido
- Proceso que es disparado en Movex por la transmisión: actualización del estado de la entrega a cliente en Movex

Cuando se anula un pedido (durante la selección de pedidos por no poderse servir algún artículo, o después de la preparación por quedar todos sus contenedores vacíos), el fichero CP generado tiene:

- ‘0000000000000000000’, en el campo Identificador Bulto
- ‘FICT ‘, en el campo Código Tipo Bulto
- 0 en las cantidades servidas de los artículos

Información transmitida

Nivel	Campo	Descripción	Tipo	Obligatorio
[H1]				
	Número Entrega	Identificador único de la entrega (corresponde con el albarán o pedido SEGA)	N15	Si
	Fecha	Fecha: AAAAMMDD	N8	Si
	Hora	Hora: HHMMSS	N6	Si
[H2]				
	Identificador Línea Movex	Identificador de la línea para Movex: contiene el texto ‘BL’	C2	Si
	Identificador Bulto	Etiqueta del bulto: número entrega (15) + número bulto (3)	C18	Si
	Código Tipo Bulto	Código del tipo de bulto	C15	Si
[H3]				
	Posición	Número de línea del pedido en Movex	C16	Si
	Código SKU	Código del artículo	C15	Si
	Cantidad	Cantidad expedida en el bulto (en packs si el tipo de pedido es normal, en unidades si el tipo de pedido es NSD)	N9	Si

3.2.3 Expediciones ('CS')

Es la confirmación a Movex de la salida (expedición) del centro de distribución de un conjunto de entregas para cliente.

- Evento que dispara la transmisión de SEGA: el cierre de una expedición
- Proceso que es disparado en Movex por la transmisión: actualización del stock, de las entregas a clientes, datos de transporte.

Cuando se anula un pedido (durante la selección de pedidos por no poderse servir algún artículo, o después de la preparación por quedar todos sus contenedores vacíos), después del fichero CP (indicando las cantidades servidas a 0), ya no se envía el fichero CS.

Información transmitida

Nivel	Campo	Descripción	Tipo	Obligatorio
[H1]				
	Número Transporte	Identificador único del transporte (correlativo de expedición)	N18	Si
	Fecha	Fecha de la expedición: AAAAMMDD	N8	Si
	Hora	Hora de la expedición: HHMMSS	N6	Si
	Transportista	Código de transportista	C15	Si
[H2]				
	Número Entrega	Identificador único de la entrega (corresponde con el albarán o pedido SEGA)	N15	Si

3.2.4 Ajustes de stock (‘AS’)

Refleja cualquier modificación al stock que no responda a la operatoria normal de entradas/salidas del almacén (ej.: robos, roturas, vencimiento, deterioro, etc.)

- Evento que dispara la transmisión de SEGA: a petición desde la pantalla cliente de importación y exportación de datos
- Proceso que es disparado en Movex por la transmisión: actualización de stock en el almacén, registro de ajustes

Información transmitida

Nivel	Campo	Descripción	Tipo	Obligatorio
[H1]				
	Fecha	Fecha del ajuste: AAAAMMDD	N8	Si
	Hora	Hora del ajuste: HHMMSS	N6	Si
	Código SKU	Código del artículo	C15	Si
	Signo	Signo del ajuste: ‘+’ o ‘ ‘: para incrementos ‘-‘: para decrementos	C1	
	Cantidad	Cantidad ajustada	N9	Si
	Tipo Unidad	Tipo de unidad en que está expresada la cantidad: ‘P’: la cantidad son packs (caso normal) ‘U’: la cantidad son unidades (caso excepcional, cuando se pierden unidades durante una ola NSD)	C1	Si
	Motivo Ajuste	Motivo del ajuste: ‘5A’: ajuste por descuadre de stock ‘5B’: ajuste por muestras sin cargo ‘5C’: ajuste por genero dañado ‘5D’: ajuste por envío a fábrica ‘5E’: ajuste por obsoleto ‘5F’: ajuste por reclasificación ‘5G’: ajuste por otros	C2	Si

3.2.5 Stock actual de artículos ('SA')

Envío de la situación actual del stock de cada uno de los artículos.

- Evento que dispara la transmisión de SEGA: periódicamente según parámetro del sistema
- Proceso que es disparado en Movex por la transmisión: comparación con el stock de Movex e impresión de un listado de discrepancias de stock

Información transmitida

Nivel	Campo	Descripción	Tipo	Obligatorio
[H1]				
	Fecha	Fecha: AAAAMMDD	N8	Si
	Hora	Hora: HHMMSS	N6	Si
	Código SKU	Código del artículo	C15	Si
	Cantidad	Cantidad total en el almacén en packs	N9	Si

3.2.6 Confirmación de mensaje ('CF')

Notificación al emisor de que un mensaje proveniente de Movex ha sido aceptado o rechazado por SEGA. Todo mensaje recibido por SEGA generará un mensaje de confirmación.

- Evento que dispara la transmisión de SEGA: el fin de tratamiento del mensaje original
- Proceso que es disparado en Movex por la transmisión: el tratamiento en Movex de los mensajes de rechazo, difícilmente podrá ser automatizado. Este mensaje deberá ser analizado por personal de sistemas para corregir el error y reenviar los mensajes de nuevo.

Información transmitida

Nivel	Campo	Descripción	Tipo	Obligatorio
[H1]				
	Nombre Mensaje Original	Nombre del fichero tratado	C25	Si
	Resultado	Resultado del tratamiento: '01': mensaje aceptado '02': error de formato del mensaje '03': fichero con código de mensaje desconocido '04': el estado de SEGA no permite su tratamiento	N2	Si