Grado en Ingeniería Informática

Trabajo de Final de Grado

Programación de módulo de fabricación inversa

Autor:
Raúl Martí Muñoz

Supervisor:
Sergi Vilar Domènech
Tutor académico:
Jorge Sales Gil

Fecha de lectura: 18 de Junio de 2024
Curso académico 2023/2024

Resumen

El siguiente documento detalla la memoria realizada para el trabajo de fin de grado del alumno Raúl Martí Muñoz, realizado en la Universidad Jaume I ubicada en Castellón, durante las prácticas en la empresa Innova Advanced Consulting S.L. La memoria contiene el proceso que se ha seguido para la realización de un módulo de fabricación inversa.

El módulo añade una nueva funcionalidad estándar de fabricación a Business Central [5], la cual es capaz de generar los componentes necesarios para la producción de un producto a partir de unas entradas generales y unas salidas de producto, además de generar todos los movimientos necesarios dentro de Business Central. También cuenta con una funcionalidad de plantillas, la cual guarda salidas generales de producto para evitar crearlas a mano.

Palabras clave

ERP, Microsoft Business Central, Orden de producción, Fabricación.

Keywords

ERP, Microsoft Business Central, Production Order, Manufacture.

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

1.1. Contexto y motivación del proyecto

En este documento se presenta la memoria de un proyecto que ha sido desarrollado en la empresa Innova Advanced Consulting[1]. Esta empresa, formada por una gran plantilla y más de veinte años en el sector, está especializada en soluciones tecnológicas y consultoría.

El proyecto consiste en realizar un módulo para el ERP (Enterprise Resource Planning) Microsoft Dynamics 365 Business Central (D365 BC)[2], ERP sobre el que trabaja la empresa Innova Advanced Consulting adaptando y creando módulos para clientes, el cual, a partir de las entradas de materia prima y las salidas de producción obtenidas, permitirá general los pedidos de ensamblado necesarios para la elaboración de los productos y distribuir consumos y costes entre los componentes de estos.

Entre las tareas que se deben implementar, se pueden destacar la realización de una nueva página en el ERP que muestre los diferentes productos obtenidos, así como la materia prima a partir de la cual se han obtenido los mismos, las diferentes opciones de modificación de esta información y los diferentes campos calculados para poder ver el coste total dividido entre los productos finales.

1.2. Objetivo y alcance del producto

El módulo creado debe ser capaz de establecer, a partir de una materia prima inicial, la parte de materia prima que pertenece a cada producto conseguido al finalizar el proceso de producción. De esta manera, se puede obtener el gasto de cada producto independiente, permitiendo a clientes del ERP que no tienen una línea de producción exacta para sus productos, sino que cada vez obtienen una cantidad de cada producto diferente con la misma cantidad de materia prima inicial, poder gestionar los datos de sus productos y facturas de forma precisa.

[1] `https://www.innovaconsulting.es`

[2] `https://www.microsoft.com/es-es/dynamics-365/products/business-central`

1.2.1. Alcance funcional

Entre las funciones principales que el sistema puede realizar se encuentran las siguientes:

- Crear plantillas de producción, las cuales pueden tener diferentes líneas de producto y componentes vinculados a las líneas de producto.
- Crear una orden de producción a partir de una plantilla.
- Introducir la materia prima que se utiliza en la orden de producción.
- Generar los componentes de cada línea de producto.
- Seleccionar la forma en que se reparte la materia prima en cada línea de producto (Especificada por línea o proporcional a salida).
- Generar los movimientos necesarios para reservar los productos en el entorno de BC.
- Generar el seguimiento del producto dentro de BC.
- Seleccionar el lote del cual se obtendrá la materia prima inicial.

1.2.2. Alcance organizacional

El alcance organizacional del módulo programado abarca a los trabajadores de la empresa que lo utilicen para gestionar la producción de sus productos.

1.2.3. Alcance informático

El módulo desarrollado se integra en uno de los módulos ya existentes en BC, añadiendo una nueva funcionalidad complementaria a la ya existente. A parte del módulo en el cual está integrado, también modifica otros módulos ya existentes:

- Productos: Accede a los datos de los productos existentes para poder seleccionar la materia prima en la orden de producción y poder generar los componentes de las líneas de producto.
- Producción: Modifica la funcionalidad estándar de las órdenes de producción para añadir una funcionalidad adicional al producto.
- Compras: Accede a los movimientos de compra para poder asignar un lote de producto a las entradas generales de materia prima.
- Gestión: Genera los movimientos de reserva en las tablas de gestión de almacenes.

1.3. Objetivo y alcance del proyecto

El objetivo principal del proyecto es desarrollar un módulo para el ERP D365 BC (Dynamics 365 Business Central) de Microsoft, añadiendo una nueva funcionalidad a la existente, la cual se encarga de calcular el precio de producción y dividirlo entre los diferentes productos obtenidos a partir de la entrada de materia prima.

El objetivo principal se puede desglosar en los siguientes subobjetivos:

- Conseguir todos los conocimientos y habilidades necesarias para realizar análisis, diseñar y desarrollar soluciones para BC partiendo de un requerimiento funcional del cliente.
- Creación de las estructuras de datos necesarias para almacenar los datos del nuevo módulo.
- Creación de una "*page part*" que muestre los campos de la tabla de entradas generales.
- Creación de una herramienta para distribuir consumos en base a las entradas.
- Creación de campos calculados para automatizar el cálculo de gastos de producción.

1.4. Descripción detallada del proyecto

Actualmente, BC es un ERP utilizado por varias organizaciones por contar con una amplia funcionalidad y cantidad de módulos diferentes, los cuales abarcan diferentes áreas entre las cuales se pueden encontrar las finanzas, compras, producción, ventas, etc. lo que lo convierte en un ERP versátil y apto para diferentes sectores.

Sin embargo, BC sigue teniendo problemas para adaptarse a algunas industrias concretas con su funcionalidad de producción, esto se debe a que su funcionalidad básica se basa en que un producto tiene asignados unos componentes ya preestablecidos para poder producirlo. Este funcionamiento no es válido para algunas industrias de alimentos, las cuales no pueden saber de forma exacta qué cantidad de cada producto se va a obtener a partir de una materia prima inicial.

Con el desarrollo de este módulo se busca añadir una funcionalidad adicional al módulo de producción de BC, mediante la cual, se permitirá repartir la materia prima inicial en los diferentes productos obtenidos al final de la producción. De esta manera, se puede repartir de forma exacta la materia prima entre los productos finales obtenidos para poder hacer un seguimiento de costes y producción detallados. Este nuevo módulo permitirá a la producción utilizar la nueva funcionalidad sin modificar el funcionamiento básico del módulo de producción de BC, permitiendo utilizar una combinación de las dos en caso de ser necesario.

Para la realización del proyecto, será necesario un analista de datos, el cual se pondrá en contacto con los clientes para estudiar su caso y poder realizar un análisis exhaustivo y una solución correspondiente a su problema. En el proyecto, también trabajará un desarrollador de software, el cual se encargará de crear el nuevo módulo e integrarlo en BC para su correcto funcionamiento.

1.4.1. Tecnologías

El módulo presentado en esta memoria se desarrollará sobre el ERP Dynamics 365 Business Central de Microsoft, el cual ofrece soluciones integradas para finanzas, contabilidad, inventario, ventas y más, principalmente a pequeñas y medianas empresas, facilitando la gestión eficiente y colaborativa de operaciones comerciales.

Como control de versiones, se utilizará GIT[3], este sistema de control de versiones se utiliza para detectar cambios en el código durante el desarrollo, de esta manera se puede trabajar de forma simultánea en diferentes ramas y seguir los diferentes cambios realizados en las mismas.

Para programar el módulo se utilizará el lenguaje AL[4], que se utiliza principalmente para personalizar y crear nuevas funcionalidades para Dynamics 365 Business Central, se trata de un lenguaje basado en eventos. Como entorno de desarrollo para AL utilizaremos Visual Studio Code[5], con un paquete que permite compilar AL.

1.5. Estructura de la memoria

En esta sección se explica cómo se organiza el resto de la memoria:

- El **capítulo 1** explica la empresa en la cual se han realizado las prácticas y el proyecto, así como su motivación, objetivos y alcance.
- El **capítulo 2** explica la metodología y la planificación previa del proyecto, así como el seguimiento de dicha planificación junto con el cálculo de los costes y los riesgos encontrados.
- El **capítulo 3** reúne los diferentes requisitos que deberá tener el producto final junto a los casos de uso necesarios para cumplirlos.
- El **capítulo 4** contiene los diferentes diseños utilizados tanto en los diferentes componentes del sistema, como en las diferentes bases de datos utilizadas por los mismos.
- El **capítulo 5** incluye la parte más práctica del proyecto, esto hace referencia a los diferentes elementos que se han desarrollado por código. También contiene las pruebas creadas para probar el sistema y el código escrito.
- Finalmente, el **capítulo 6** contiene una reflexión sobre el proyecto realizado en la memoria y los diferentes conocimientos que se han adquirido con la realización del mismo.

[3] `https://git-scm.com`

[4] AL es el lenguaje de programación utilizado para manipular datos (recuperar, insertar y modificar registros) en una base de datos de Dynamics 365 Business Central.

[5] `https://code.visualstudio.com`

Capítulo 2

Planificación del proyecto

2.1. Metodología

En este proyecto se hará uso de una metodología predictiva [3], la cual se basa en planificar y prever los distintos aspectos del proyecto. Precisamente, dado que se sabe exactamente a dónde se quiere llegar y cómo se va a hacer el proyecto, esta metodología consigue llegar al final de manera rápida y segura.

Este proyecto se dividirá en las siguientes fases:

- Iniciación: Se realiza un estudio del proyecto, con el cual se establecerá la idea de este, así como su alcance, objetivos y riesgos.
- Planificación: Una vez terminado el esbozo inicial del proyecto, se realiza una estimación temporal y de costes del proyecto junto a un diagrama de Gantt inicial que se puede observar en la Figura 2.2.
- Formación: Una vez terminada la planificación, se realiza un estudio sobre Microsoft Dynamics 365 Business Central para familiarizarse con el entorno. Para complementar la formación sobre el entorno, se realiza una formación sobre AL para poder programar el proyecto.
- Análisis: Se especifican más concretamente las funcionalidades que va a tener el producto final y como sé van a gestionar los requisitos que debe cumplir el proyecto.
- Ejecución: Una vez terminada la toda la planificación inicial del producto se procede a diseñar la base de datos y la interfaz que va a tener el proyecto, también se empieza a programar el producto final que se entregará al cliente.
- Pruebas: Finalmente, se requiere crear una test-app para poder probar si el producto final funciona de la manera adecuada y esperada.

2.2. Planificación temporal del proyecto

Para la planificación temporal del proyecto, se ha creado el diagrama de estructura de descomposición de trabajo de la Figura 2.1, el cual contiene las fases anteriormente nombradas.

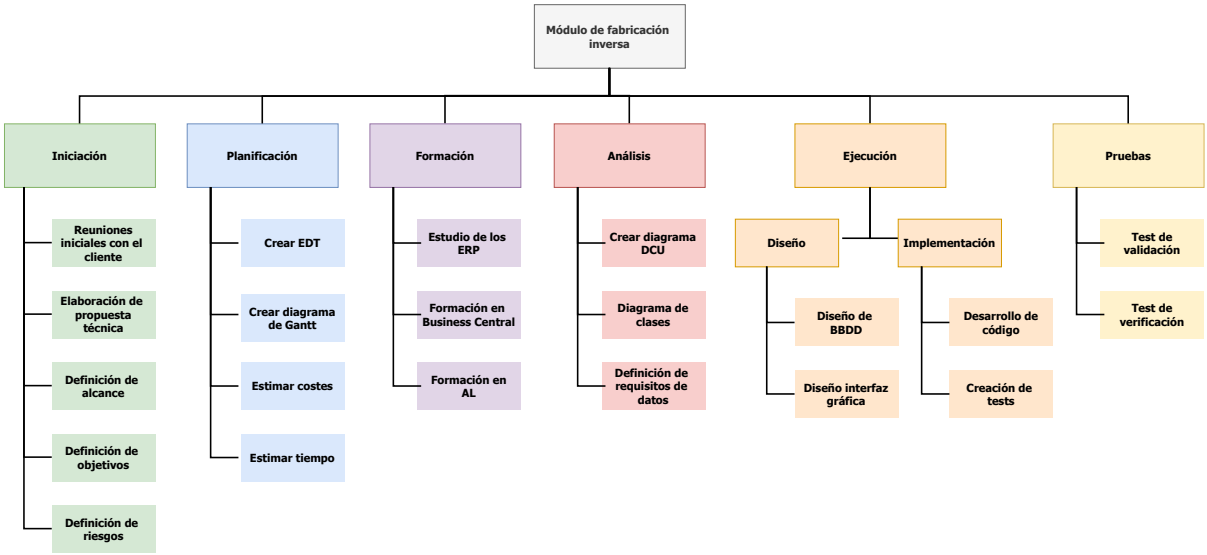

Figura 2.1: Diagrama de estructura de descomposición de trabajo.

A la hora de estimar el tiempo del proyecto, no se han incluido las dos primeras fases del proyecto, solo se han tenido en cuenta las fases de formación, análisis, ejecución y pruebas. Dichas fases empiezan el día 05/02/2024 y terminan el día 10/04/2024, haciendo una jornada de siete horas diarias durante cinco días a la semana, hacen una duración total de 300 horas que ha necesitado el proyecto para finalizarse. En el diagrama de Gantt que se puede ver en la Figura 2.2, podemos observar cómo se distribuyen las 300 horas de proyecto en las diferentes tareas creadas, teniendo en cuenta que el diseño y la implementación se han realizado de forma paralela.

2.3. Seguimiento del proyecto

Una de las partes principales de este proyecto ha sido llevar a cabo un seguimiento bien organizado y detallado, esto se debe al límite de tiempo ajustado que se ha tenido para realizarlo. Para llevar a cabo este seguimiento, se han realizado informes quincenales sobre el progreso del proyecto para que el tutor de prácticas estuviese informado del avance en todo momento.

Id	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
1	**Módulo de fabricación inversa**	**300 hora**	**lun 05/02/24**	**mié 10/04/24**	
2	**Formación**	**126 hora**	**lun 05/02/24**	**mié 28/02/24**	
6	**Análisis**	**21 horas**	**jue 29/02/24**	**mar 05/03/24**	
7	Definicion de requisitos	7 horas	jue 29/02/24	jue 29/02/24	5
8	Análisis de requisitos	7 horas	vie 01/03/24	vie 01/03/24	7
9	Análisis de funcionalidad	7 horas	mar 05/03/24	mar 05/03/24	8
10	**Ejecución**	**112 hora**	**mié 06/03/24**	**mar 02/04/24**	
11	**Entradas generales**	**14 horas**	**mié 06/03/24**	**jue 07/03/24**	
12	Creación de estructura de datos	7 horas	mié 06/03/24	mié 06/03/24	9
13	Modificación de vista de ordenes de producción	7 horas	jue 07/03/24	jue 07/03/24	12
14	**Plantillas**	**42 horas**	**vie 08/03/24**	**vie 15/03/24**	
15	**Orden de producción**	**7 horas**	**vie 08/03/24**	**vie 08/03/24**	
16	Modificación de estructura de datos de ordenes de	3 horas	vie 08/03/24	vie 08/03/24	13
17	Modificación de vista de ordenes de producción	4 horas	vie 08/03/24	vie 08/03/24	16
18	**Página de plantilla**	**7 horas**	**lun 11/03/24**	**lun 11/03/24**	
19	Creación de estructura de datos de plantilla	3 horas	lun 11/03/24	lun 11/03/24	17
20	Creación de vista de plantilla	4 horas	lun 11/03/24	lun 11/03/24	19
21	**Lista de plantillas**	**14 horas**	**mar 12/03/24**	**mié 13/03/24**	
22	Creación estructura de datos de lista de plantillas	3 horas	mar 12/03/24	mar 12/03/24	20
23	Creación de vista de lista de plantillas	4 horas	mar 12/03/24	mar 12/03/24	22
24	Enlazar lista de plantillas con la orden de producción	7 horas	mié 13/03/24	mié 13/03/24	23
25	**Líneas de plantilla**	**7 horas**	**jue 14/03/24**	**jue 14/03/24**	
26	Creación de estructura de datos de líneas de plantilla	3 horas	jue 14/03/24	jue 14/03/24	24
27	Modificación de vista de página de plantilla	4 horas	jue 14/03/24	jue 14/03/24	26
28	**Componentes de plantilla**	**7 horas**	**vie 15/03/24**	**vie 15/03/24**	
29	Creación de estructura de datos de componentes de plantilla	3 horas	vie 15/03/24	vie 15/03/24	27
30	Creación de vista de componentes de plantilla	4 horas	vie 15/03/24	vie 15/03/24	29
31	**Producción automática**	**56 horas**	**mié 20/03/24**	**mar 02/04/24**	
32	**Generación de componentes**	**35 horas**	**mié 20/03/24**	**mar 26/03/24**	
33	Proporcional a salida	21 horas	mié 20/03/24	vie 22/03/24	30
34	Especificado por línea	14 horas	lun 25/03/24	mar 26/03/24	33
35	**Aplicación de plantilla**	**21 horas**	**mié 27/03/24**	**mar 02/04/24**	
36	Insertar líneas de salida al cambiar de plantilla	14 horas	mié 27/03/24	jue 28/03/24	34
37	Insertar componentes al cambiar de plantilla	7 horas	mar 02/04/24	mar 02/04/24	36
38	**Pruebas**	**41 horas**	**mié 03/04/24**	**mié 10/04/24**	
39	Pruebas de aceptacion	21 horas	mié 03/04/24	vie 05/04/24	37
40	Pruebas CI/CD	20 horas	lun 08/04/24	mié 10/04/24	39

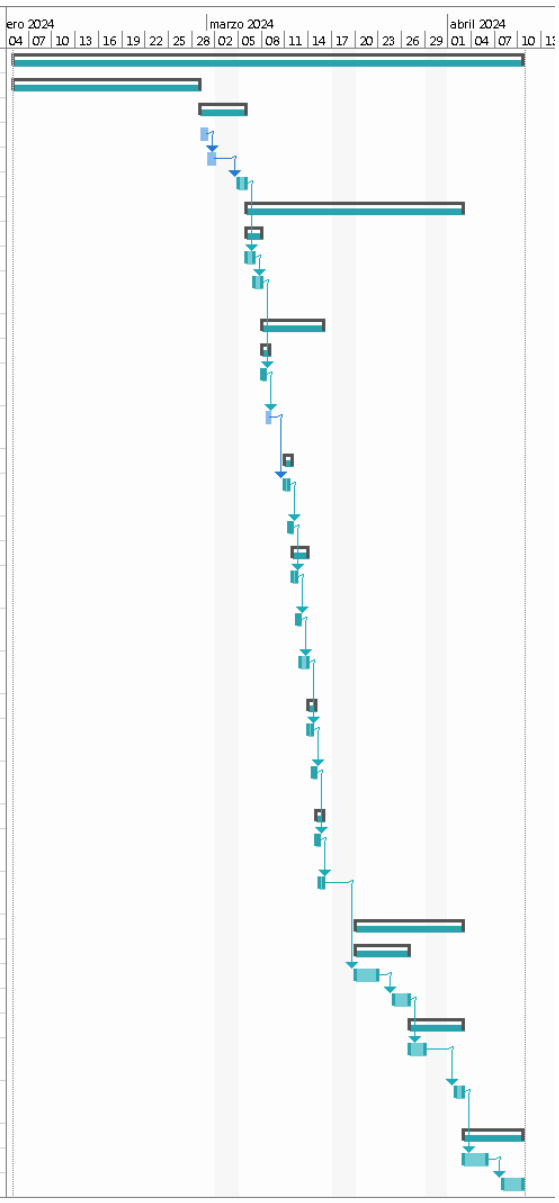

Figura 2.2: Diagrama de Gantt que representa la organización del proyecto con el desglose de tareas y su planificación temporal.

Primera quincena

La primera quincena se dedicó principalmente al estudio y comparación de los diferentes ERP del mercado, para hacer esto, en los primeros días se eligieron tres ERP para poder comparar sus principales características y precios, así como preparar una exposición sobre ellos. También se estudió el avance del ERP Microsoft Dynamics Business Central a lo largo de los años, para así entender las tecnologías que utiliza y cómo se ha adaptado al paso del tiempo.

Una vez terminado el estudio de los ERP, empezó la formación funcional, en la cual apren-

dimos a utilizar las características principales de Microsoft Dynamics Business Central para en un futuro poder programar módulos para este ERP de forma más eficiente.

Segunda quincena

Durante esta quincena se realizó la parte de formación técnica sobre AL, el lenguaje utilizado para programar módulos en Microsoft Dynamics Business Central, en la cual aprendí sobre los diferentes objetos que se pueden utilizar en el lenguaje, así como las diferentes características que tiene cada uno de ellos.

Para esta formación, se ha utilizado Visual Studio Code como entorno de programación de AL, el cual es un editor de texto que puede instalar varias extensiones para poder ser utilizado como entorno de programación.

Tercera quincena

Durante esta quincena terminé la formación técnica en los últimos elementos del lenguaje AL. Una vez terminada esta formación, leí la documentación del módulo para el ERP Microsoft Dynamics Business central, en la cual explica los diferentes elementos que contiene el mismo para poder realizarlo detalladamente.

Una vez leída la documentación del módulo, creé las diferentes tablas para guardar toda la información necesaria, así como modificar tablas a través de diferentes *TableExtension*, las cuales permiten añadir y modificar tablas ya existentes en el núcleo de BC.

Una vez preparadas las tablas, he empezado a crear las *Page*, así como modificar una *Page* ya existente para poder añadir las tablas que he creado anteriormente. También uní las tablas anteriores con las *Page*, de esta forma el usuario puede acceder a la información del módulo.

En cuanto a las funcionalidades, añadí la funcionalidad básica del módulo, la cual se ha acompañado con la creación de diferentes plantillas para que el usuario pueda crear datos predefinidos en el módulo.

Cuarta quincena

Durante esta quincena se mejoró el cálculo de cantidades de la funcionalidad estándar junto a controlar posibles errores del usuario a la hora de utilizarla.

Se actualizó por completo la funcionalidad de plantillas para las órdenes de producción, las cuales serán capaces de almacenar líneas de producto con sus respectivos componentes para que al usuario se le completen estos campos automáticamente al seleccionar una plantilla para la orden de producción. En estas plantillas también se ha añadido la posibilidad de cambiar la forma de calcular las cantidades para las entradas de líneas generales, para hacer esto la

plantilla tendrá un desplegable en el cual se podrá elegir si se quiere calcular "Especificado por línea" o "Proporcional a salida".

Para no interferir con la funcionalidad estándar de BC he creado diferentes "*CodeUnits*"[1], los cuales crean todas las entradas necesarias en la base de datos de BC para que luego éste pueda funcionar sin problemas de datos cuando se lance la orden de producción.

Finalmente, realicé la aplicación de test para probar el correcto funcionamiento del módulo e integré la implementación continua con GIT para que se comprueben los test en cada *"commit"* que se crea. Esta funcionalidad también permite probar el código sin tener un entorno de BC disponible en ese momento, ya que crea una virtual para probarlo desde GIT.

2.4. Costes

Para este proyecto se espera hacer una estimación de los recursos y costes que se han necesitado para completar el proyecto en la empresa.

2.4.1. Recursos de Hardware

Inicialmente se han adquirido herramientas de trabajo para el desarrollo del proyecto. Estas herramientas se componen de diferentes periféricos y un ordenador portátil, los cuales suman un coste total de 655 €, los precios separados son los siguientes:

- Ordenador portátil: 550 €.
- Monitor adicional: 75 €.
- Ratón y teclado: 30 €.

Para calcular el precio que supondrán estos elementos en nuestro proyecto, debemos adaptarlo al tiempo que se ha tardado en desarrollar el proyecto. En la Tabla 2.1 podemos ver que los costes de hardware de este proyecto serían 23,51 €.

Hardware	**Precio (€)**	**Vida útil (años)**	**66 días (€)**
Portatil	550,00	5	19,89
Monitor	75,00	6	2,26
Teclado y ratón	30,00	4	1,36
			23,51

Cuadro 2.1: Resumen de costes de hardware.

[1]Un *CodeUnit* es un contenedor de código AL que se puede utilizar en muchos objetos de aplicación.

2.4.2. Recursos de Software

Para poder desarrollar este proyecto se han utilizado diversos programas necesarios para poder crear el módulo para BC, pero solo tendremos en cuenta el coste de aquellos utilizados específicamente para el proyecto.

- Visual Studio Code: Ya que se trata de software de código abierto, se utilizará la versión gratuita para el proyecto.
- GitHub: En este proyecto se utilizará la versión gratuita de GitHub, ya que se adapta adecuadamente a las diferentes necesidades del proyecto.
- Dynamics 365 Business Central: Para poder desarrollar el proyecto en este ERP se necesita la funcionalidad completa del mismo, por ello, se necesitará una licencia de pago la cual tiene un precio de 65,50 € al mes.

La duración de este proyecto será de 66 días, por lo cual se debe calcular el precio para dicha duración. En la Tabla 2.2 podemos ver que los costes totales de software pare este proyecto serían de 196,50 €.

Software	**Precio/Mes (€)**	**66 días**
Visual studio code	0,00	0,00
GitHub	0,00	0,00
D365 Business Central	65,50	196,50
		196,50

Cuadro 2.2: Resumen de costes de software.

2.4.3. Recursos humanos

Este proyecto se ha realizado únicamente por una persona, la cual ocupa diferentes roles a lo largo del proyecto.

- Analista: Inicialmente, realiza el rol de analista de datos, por lo cual se encarga de investigar las necesidades iniciales del proyecto y el cliente, a través de las cuales diseña una solución. Como hemos visto anteriormente en la Figura 2.2, ejerce un total de 21 horas en este rol.
- Desarrollador: Una vez se han investigado las necesidades iniciales del proyecto, se cambia al rol de desarrollador, en el cual recibirá una formación de 126 horas y realizará el desarrollo del producto, que ocupará un total de 112 horas junto a las pruebas, que tarda en realizarlas 41 horas. En conjunto hacen un total de 279 horas.

Para calcular los costes de recursos humanos, se ha obtenido el sueldo medio de los analistas y desarrolladores en BC [7]. Con esta información se ha obtenido el coste total de cada rol

dependiendo de las horas que se han invertido en cada uno de los mismos. Esta información resumida se puede consultar en la Tabla 2.3.

Rol	**Sueldo/hora (€)**	**Horas totales**	**Total (€)**
Analista	32,00	21	672,00
Desarrollador	16,92	279	4.720,68
			5.392,68

Cuadro 2.3: Resumen de costes de recursos humanos.

Una vez obtenidos los costes directos de los recursos humanos, también se deben tener en cuenta los costes indirectos y diferentes gastos adicionales de contratación. Los costes de contratación suelen estar en un 25 % de los costes de recursos humanos y los costes indirectos un 20 %. De esta manera obtenemos un coste total de recursos humanos de **7.819,39 €.**

2.4.4. Resumen de costes

Teniendo en cuenta los costes anteriores y realizando un sumatorio de ellos, podemos obtener que el proyecto tendrá un coste próximo a **8.039,40 €**, como se puede ver en la Tabla 2.4.

Costes	**Total (€)**
Software	196,50
Hardware	23,51
Humanos	7.819,39
	8.039,40

Cuadro 2.4: Resumen de costes totales del proyecto.

2.5. Riesgos

En esta sección se van a numerar los diferentes riesgos que han aparecido a lo largo del proyecto, así como el plan de contención y contingencia de cada uno.

2.5.1. Análisis de riesgos

Inicialmente, vamos a nombrar los diferentes riesgos que ha habido en el proyecto, indicando la magnitud que han tenido sobre el proyecto, una descripción de ellos, el impacto que pueden tener sobre el proyecto y el tipo de riego. Toda esta información se puede consultar en la Tabla 2.5.

Id	Análisis
R01	**Riesgo:** Tiempo limitado. **Magnitud:** Alta. **Descripción:** El tiempo limitado de 300 horas en este proyecto puede suponer un grave problema, ya que en caso de que aparezca algún fallo o retraso inesperado deja poco margen de acción para terminarlo en el periodo acordado. **Impacto:** El proyecto puede tardar más tiempo del esperado en terminar. **Tipo:** Riesgo de proyecto.
R02	**Riesgo:** Inexperiencia en ERP. **Magnitud:** Media. **Descripción:** El desarrollador del proyecto no tiene experiencia previa en la utilización del ERP, lo cual puede llegar a influir negativamente en el progreso del proyecto. **Impacto:** Puede que el producto no se haga siguiendo el estandar de BC. **Tipo:** Riesgo de producto.
R03	**Riesgo:** Inexperiencia en AL. **Magnitud:** Muy alta. **Descripción:** El desconocimiento del lenguaje de programación AL, el cual se utilizará para programar el módulo para el ERP, es el reto más importante en este proyecto. **Impacto:** El proyecto puede tardar más tiempo del esperado en terminar. **Tipo:** Riesgo de proyecto y producto.
R04	**Riesgo:** Dificultad de entendimiento de requisitos. **Magnitud:** Media. **Descripción:** Los requisitos de este proyecto son de una dificultad técnica, dentro del funcionamiento estándar de BC y su entorno de producción, bastante elevada. Esto puede suponer un problema a la hora de crear el proyecto ya, que puede haber malentendidos al añadir la nueva funcionalidad. **Impacto:** El producto final puede no ser el esperado por el cliente. **Tipo:** Riesgo de producto y proyecto.

Cuadro 2.5: Análisis de riesgos.

2.5.2. Prevención de riesgos

Una vez analizados los diferentes riesgos que tiene este proyecto, se ha de crear una solución para los mismos, de esta manera se crean planes de prevención y contingencia para cada uno de ellos.

R01

- Plan de prevención: Realizar un análisis exhaustivo del proyecto, de esta manera se podrá realizar una planificación temporal adecuada para adaptarse al tiempo limitado.
- Plan de contingencia: Eliminar funcionalidades secundarias que nos son necesarios para el funcionamiento básico del proyecto.

R02

- Plan de prevención: Realizar un estudio de mercado inicial sobre los diferentes ERP, seguido de una formación sobre el ERP BC, principalmente los módulos que se van a utilizar para la realización del proyecto.
- Plan de contingencia: Contactar constantemente un especialista en el ERP BC para que pueda revisar el proyecto continuamente.

R03

- Plan de prevención: Realizar una formación inicial sobre el lenguaje AL, la cual será impartida por una persona con experiencia previa en el lenguaje. Esta persona será un integrante de la empresa.
- Plan de contingencia: Realizar pequeñas formaciones adicionales del lenguaje en caso de que sea necesario adquirir conocimientos adicionales sobre el lenguaje AL.

R04

- Plan de prevención: Realizar un análisis exhaustivo del proyecto, de esta manera dejar bien claro con el cliente todos los puntos que quiere abarcar el proyecto y como debe ser el resultado final del mismo.
- Plan de contingencia: Realizar reuniones frecuentes con el cliente para que pueda revisar el avance del proyecto y ver si se adapta a su idea inicial.

Capítulo 3

Análisis del sistema/producto

3.1. Definición de requisitos

Para identificar los diferentes requisitos del sistema, se ha llevado a cabo un estudio funcional del mismo, de esta manera se han especificado los objetivos del proyecto y las diferentes funcionalidades que debe otorgar el mismo a los trabajadores. Se ha llegado a la conclusión de que la forma que más adapta para definir los requisitos es mediante un diagrama de casos de uso, en el cual podremos ver los siguientes:

- CU_01: Gestionar plantillas.
- CU_02: Gestionar líneas de plantilla.
- CU_03: Gestionar componentes de líneas de plantilla.
- CU_04: Añadir entradas generales.
- CU_05: Generar componentes.
- CU_06: Gestionar líneas de una orden de producción.

3.1.1. Diagrama de casos de uso

El diagrama de casos de uso nos permite identificar los diferentes actores que tiene el sistema, así como las diferentes funcionalidades del mismo que podrá utilizar cada actor. En este caso, como se puede ver en la Figura 3.1, el diagrama es bastante simple dada la funcionalidad directa que tiene el sistema con el actor.

En el diagrama se puede apreciar un único actor, esto se debe a que la funcionalidad completa del proyecto es utilizada únicamente por un usuario que se encargará de generar las órdenes de producción iniciales.

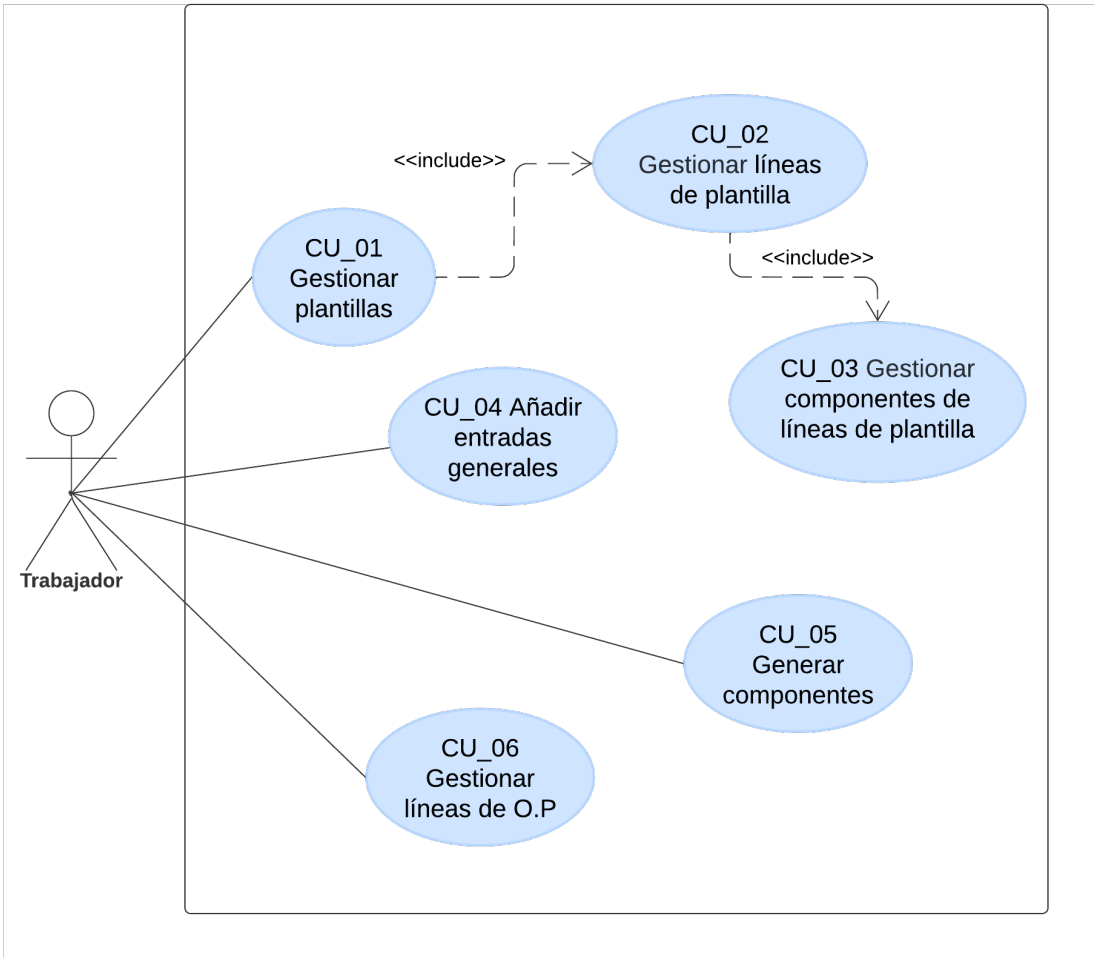

Figura 3.1: Diagrama de casos de uso [2].

3.1.2. Especificación de casos de uso

A continuación se detallan los diferentes casos de uso que se pueden observar en la Figura 3.1, nombrando una descripción de los mismos y diferentes secuencias que se pueden seguir para realizarlos.

Identificador	CU_01
Nombre:	Gestionar plantillas
Actor:	Trabajador
Descripción:	Este caso de uso representa la funcionalidad de crear, modificar y eliminar plantillas para una orden de producción.
Secuencia normal:	Se desea crear una plantilla. 1. El trabajador accede a las plantillas de orden de producción seleccionando “Plantillas orden de producción”. 2. El trabajador selecciona la opción “Nuevo” en el menú superior. 3. El trabajador rellena los diferentes campos de la plantilla (Nº, Nombre, Descripción y Aplicación de costes). 4. El sistema guarda la plantilla en la base de datos.
Secuencia normal:	Se desea modificar una plantilla. 1. El trabajador accede a las plantillas de orden de producción seleccionando “Plantillas orden de producción”. 2. El trabajador selecciona una plantilla de las disponibles en el sistema. 3. El trabajador modifica uno o varios de los campos de la plantilla (Nº, Nombre, Descripción y Aplicación de costes). 4. El sistema guarda los cambios de la plantilla en la base de datos.
Secuencia normal:	Se desea eliminar una plantilla 1. El trabajador accede a las plantillas de orden de producción seleccionando “Plantillas orden de producción”. 2. El trabajador selecciona una plantilla de las disponibles en el sistema. 3. El trabajador selecciona la opción “Borrar” en el menú superior. 4. Se elimina la plantilla de la base de datos.
Secuencia alternativa:	Se desea modificar una plantilla. 1. El trabajador accede a las plantillas de orden de producción seleccionando “Plantillas orden de producción”. 2. El trabajador selecciona una plantilla de las disponibles en el sistema. 3. El trabajador selecciona la opción “Editar” del menú superior. 4. El trabajador modifica uno o varios de los campos de la plantilla (Nº, Nombre, Descripción y Aplicación de costes). 5. El sistema guarda los cambios de la plantilla en la base de datos.
Excepciones	1. El trabajador intenta añadir una plantilla con un “Nº” que ya existe. 2. El trabajador intenta añadir una plantilla con el cambo “Nº” en blanco.

Cuadro 3.1: Detalles del caso de uso CU_01.

Identificador	CU_02
Nombre:	Gestionar líneas de plantilla
Actor:	Trabajador
Descripción:	Este caso de uso representa la funcionalidad de crear, modificar y eliminar líneas de plantilla para una plantilla concreta.
Secuencia normal:	Se desea añadir una línea de salida a una plantilla. 1. El trabajador accede a las plantillas de orden de producción seleccionando "Plantillas orden de producción". 2. El trabajador selecciona una plantilla de las disponibles en el sistema. 3. El trabajador selecciona una de las líneas vacías en el campo líneas. 4. El trabajador rellena los campos de la línea de la plantilla (Nº producto, Descripción de producto y Aplicación de costes %). 5. El sistema guarda los cambios de la plantilla en la base de datos.
Secuencia normal:	Se desea modificar una línea de salida a una plantilla. 1. El trabajador accede a las plantillas de orden de producción seleccionando "Plantillas orden de producción". 2. El trabajador selecciona una plantilla de las disponibles en el sistema. 3. El trabajador selecciona una de las líneas vacías en el campo líneas. 4. El trabajador modifica uno de los campos de la línea de salida. 5. El sistema guarda los cambios de la plantilla en la base de datos.
Secuencia normal:	Se desea eliminar una línea de salida a una plantilla. 1. El trabajador accede a las plantillas de orden de producción seleccionando "Plantillas orden de producción". 2. El trabajador selecciona una plantilla de las disponibles en el sistema. 3. El trabajador selecciona una de las líneas vacías en el campo líneas. 4. El trabajador selecciona la opción "Borrar línea" del menú superior. 5. Se elimina la línea de salida de la base de datos.
Secuencia alternativa:	Se desea eliminar una línea de salida a una plantilla. 1. El trabajador accede a las plantillas de orden de producción seleccionando "Plantillas orden de producción". 2. El trabajador selecciona una plantilla de las disponibles en el sistema. 3. El trabajador selecciona una de las líneas vacías en el campo líneas. 4. El trabajador selecciona los tres puntos de la línea, aparece un menú desplegable donde se selecciona la opción "Borrar línea". 5. Se elimina la línea de salida de la base de datos.
Excepciones	1. El trabajador intenta añadir una línea de salida sin producto.

Cuadro 3.2: Detalles del caso de uso CU_02.

Identificador	CU_03
Nombre:	Gestionar componentes de líneas de plantilla
Actor:	Trabajador
Descripción:	Este caso de uso representa la funcionalidad de crear, modificar y eliminar componentes de una línea de plantilla.
Secuencia normal:	Se desea añadir un componente a una línea de salida de una plantilla. 1. El trabajador accede a las plantillas de orden de producción seleccionando "Plantillas orden de producción". 2. El trabajador selecciona una plantilla de las disponibles en el sistema. 3. El trabajador selecciona una de las líneas de plantilla en el campo líneas. 4. El trabajador selecciona el menú "Línea" del apartado Línea. 5. El trabajador selecciona "Componentes" en el menú superior. 6. El trabajador selecciona una línea de la lista de componentes de plantilla. 7. El trabajador rellena los campos de la línea seleccionada (Nº producto, Descripción de producto). 8. El sistema guarda los cambios de los componentes en la base de datos.
Secuencia normal:	Se desea modificar un componente de una línea de salida de una plantilla. 1. El trabajador accede a las plantillas de orden de producción seleccionando "Plantillas orden de producción". 2. El trabajador selecciona una plantilla de las disponibles en el sistema. 3. El trabajador selecciona una de las líneas de plantilla en el campo líneas. 4. El trabajador selecciona el menú "Línea" del apartado Línea. 5. El trabajador selecciona "Componentes" en el menú superior. 6. El trabajador selecciona una línea de la lista de componentes de plantilla. 7. El trabajador modifica los campos de la línea seleccionada. 8. El sistema guarda los cambios de los componentes en la base de datos.
Secuencia normal:	Se desea eliminar un componente de una línea de salida de una plantilla. 1. El trabajador accede a las plantillas de orden de producción seleccionando "Plantillas orden de producción". 2. El trabajador selecciona una plantilla de las disponibles en el sistema. 3. El trabajador selecciona una de las líneas de plantilla en el campo líneas. 4. El trabajador selecciona el menú "Línea" del apartado Línea. 5. El trabajador selecciona "Componentes" en el menú superior. 6. El trabajador selecciona una línea de la lista de componentes de plantilla. 7. El trabajador selecciona "Eliminar" en el menú superior. 8. Se elimina el componente de la línea de salida de la plantilla.
Excepciones:	1. El trabajador introduce un componente sin el campo "Nº producto".

Cuadro 3.3: Detalles del caso de uso CU_03.

Identificador	CU_04
Nombre:	Añadir entradas generales
Actor:	Trabajador
Descripción:	Este caso de uso representa la funcionalidad de crear, modificar y eliminar entradas generales en una orden de producción.
Secuencia normal:	Se desea añadir una entrada general a una orden de producción planificada. 1. El trabajador accede a las órdenes de producción planificadas seleccionando "Órdenes de producción planificadas". 2. El trabajador selecciona una orden de producción de la lista. 3. El trabajador accede al apartado "Entradas generales" de la ventana. 4. El trabajador selecciona una línea vacía. 5. El trabajador rellena los campos de la línea seleccionada (Nº producto, Descripción de producto, Tipo, Localización, Cód. unidad de medida, Cantidad, Nº.mov, NºLote y Cantidad de lote). 6. El sistema guarda la nueva entrada general en la base de datos.
Secuencia normal:	Se desea modificar una entrada general de una orden de producción planificada. 1. El trabajador accede a las órdenes de producción planificadas seleccionando "Órdenes de producción planificada". 2. El trabajador selecciona una orden de producción de la lista. 3. El trabajador accede al apartado "Entradas generales" de la ventana. 4. El trabajador selecciona una línea existente. 5. El trabajador modifica uno o varios de los campos de la línea seleccionada. 6. El sistema guarda la nueva entrada general en la base de datos.
Secuencia normal:	Se desea eliminar una entrada general de una orden de producción planificada. 1. El trabajador accede a las órdenes de producción planificadas seleccionando "Órdenes de producción planificada". 2. El trabajador selecciona una orden de producción de la lista. 3. El trabajador accede al apartado "Entradas generales" de la ventana. 4. El trabajador selecciona una línea existente. 5. El trabajador selecciona "Borrar línea" en el menú superior. 6. Se elimina la entrada general de la base de datos.
Excepciones:	1. El trabajador añade una cantidad superior a la actual en el lote. 2. El trabajador crea una entrada general con el campo "Nº producto" vacio. 3. El trabajador elige una unidad de medida que no existe en la base de datos. 4. El trabajador selecciona una localización que no existe en la base de datos. 5. El trabajador selecciona un lote que no está en la localización seleccionada.

Cuadro 3.4: Detalles del caso de uso CU_04.

Identificador	CU_06
Nombre:	Gestionar líneas de orden de producción
Actor:	Trabajador
Descripción:	Este caso de uso representa la funcionalidad de crear, modificar y eliminar líneas de salida de una orden de producción, así como sus componentes.
Secuencia normal:	Se desea añadir una línea de salida a una orden de producción planificada. 1. El trabajador accede a las órdenes de producción planificadas seleccionando "Órdenes de producción planificada". 2. El trabajador selecciona una orden de producción de la lista. 3. El trabajador accede al apartado "Líneas" de la ventana. 4. El trabajador selecciona una línea vacía. 5. El trabajador rellena los campos de la línea seleccionada (Nº producto, Descripción, Nº L.M. producción, Nº ruta, Fecha-hora inicial, Fecha-hora final, Cantidad, Cód. unidad medida, Coste unitario y importe coste). 6. El sistema guarda la nueva línea de salida en la base de datos.
Secuencia normal:	Se desea modificar una línea de salida de una orden de producción planificada. 1. El trabajador accede a las órdenes de producción planificadas seleccionando "Órdenes de producción planificada". 2. El trabajador selecciona una orden de producción de la lista. 3. El trabajador accede al apartado "Líneas" de la ventana. 4. El trabajador selecciona una línea existente. 5. El trabajador modifica uno o varios de los campos de la línea. 6. El sistema guarda la nueva línea de salida en la base de datos.
Secuencia normal:	Se desea eliminar una línea de salida de una orden de producción planificada. 1. El trabajador accede a las órdenes de producción planificadas seleccionando "Órdenes de producción planificada". 2. El trabajador selecciona una orden de producción de la lista. 3. El trabajador accede al apartado "Líneas" de la ventana. 4. El trabajador selecciona una línea existente. 5. El trabajador accede al menú "Administrar" de la sección "Líneas". 6. El trabajador selecciona la opción "Borrar línea". 7. Se borra la línea de salida de la base de datos.
Excepciones:	1. El trabajador añade un producto que no existe. 2. El trabajador selecciona una unidad de medida que no existe en el sistema.

Cuadro 3.5: Detalles del caso de uso CU_06.

Identificador	CU_05
Nombre:	Generar componentes
Actor:	Trabajador
Descripción:	Este caso de uso representa la funcionalidad por la cual se generan los diferentes componentes en las líneas de salida de una orden de producción.
Secuencia normal:	Se desean generar los componentes de una línea de salida a partir de las entradas generales. 1. El trabajador accede a las órdenes de producción planificadas seleccionando "Órdenes de producción planificadas". 2. El trabajador selecciona una orden de producción de la lista. 3. El trabajador accede al apartado "Entradas generales" de la ventana. 4. El trabajador cambia al menú "Funciones" del apartado entradas generales. 5. El trabajador selecciona "Generar componentes" en el menú superior. 6. El sistema genera todos los componentes necesarios de las líneas de salida.
Excepciones:	1. El trabajador genera componentes con los campos de la entrada general incorrectos. 2. El trabajador genera los componentes sin entradas generales existentes.

Cuadro 3.6: Detalles del caso de uso CU_05.

3.1.3. Requisitos de datos

En esta sección se especificarán los diferentes requisitos de datos que serán introducidos por el trabajador en el sistema para poder realizar los diferentes procesos nombrados en los casos de uso del apartado anterior. Debido a que la base de datos está integrada directamente en BC, existen datos utilizados en los casos de uso que están contenidos en el programa base de BC, para diferenciar dichos datos los dividiremos en dos grupos diferentes:

- **Módulo:** Requisitos de datos que se han insertado, junto al módulo, en el sistema.

- **Base:** Requisitos de datos que ya existen en la base de BC, pero también son utilizados por el módulo.

Identificador	RD_01
Nombre:	Plantilla
Tipo:	Módulo
Descripción:	Requisito que representa los datos necesarios para la creación de una plantilla.
Casos de uso:	CU_01, CU_02, CU_03, CU_05
Datos específicos:	Nº, Nombre, Descripción, Aplicación de costes.
Comentarios:	**Nº:** Se especificará el número de serie de la plantilla. Este número relaciona la plantilla con sus líneas y componentes. **Nombre:** Especifica el nombre que se le quiere dar a la plantilla. **Descripción:** Especifica una descripción más detallada de la utilidad de la plantilla. **Aplicación de costes:** Especifica la forma en la que se generarán los componentes de las órdenes de producción que contengan una plantilla.

Cuadro 3.7: Detalles del requisito de datos RU_01.

Identificador	RD_02
Nombre:	Línea de salida de plantilla
Tipo:	Módulo
Descripción:	Requisito que representa los datos necesarios para la creación de una línea de salida que pertenece a una plantilla.
Casos de uso:	CU_02, CU_03, CU_05
Datos específicos:	Nº plantilla, Nº producto, Nº línea, Descripción de producto, % Aplicación de costes.
Comentarios:	**Nº plantilla:** Se especificará el número de serie de la plantilla. Este número relaciona la plantilla con sus líneas y componentes. **Nº producto:** Se seleccionará un producto de la lista de productos disponibles en el sistema. **Nº línea:** Especifica el número que identifica a dicha línea dentro de la plantilla. **Descripción de producto:** Especifica una descripción más detallada del producto seleccionado. **% Aplicación de costes:** Especifica el porcentaje de las entradas generales de una orden de producción que se adjudicarán a una línea de salida.

Cuadro 3.8: Detalles del requisito de datos RU_02.

Identificador	RD_03
Nombre:	Componentes de línea de salida de plantilla
Tipo:	Módulo
Descripción:	Requisito que representa los datos necesarios para la creación de un componente de una línea de salida que pertenece a una plantilla.
Casos de uso:	CU_03
Datos específicos:	Nº plantilla, Nº línea plantilla, Nº producto, Nº línea, Descripción de producto, % Aplicación de costes.
Comentarios:	**Nº plantilla:** Se especificará el número de serie de la plantilla. Este número relaciona la plantilla con sus líneas y componentes. **Nº línea plantilla:** Se especificará el número de serie de la línea de salida. Este número relaciona la línea de salida con sus componentes. **Nº producto:** Se seleccionará un producto de la lista de productos disponibles en el sistema. **Nº línea:** Especifica el número que identifica a dicho componente dentro de la línea de salida. **Descripción de producto:** Especifica una descripción más detallada del producto seleccionado.

Cuadro 3.9: Detalles del requisito de datos RU_03.

Identificador	RD_04
Nombre:	Línea de salida
Tipo:	Módulo
Descripción:	Requisito que representa los datos necesarios para la creación de una entrada general.
Casos de uso:	CU_06
Datos específicos:	Nº orden producción, Nº línea, Nº producto, Descripción de producto, Unidad de medida, Cantidad.
Comentarios:	**Nº orden producción:** Se especificará el número de serie de la orden de producción a la cual pertenece la línea de salida. **Nº línea:** Especifica el número que identifica la línea de salida. **Nº producto:** Se seleccionará un producto de la lista de productos disponibles en el sistema. **Descripción de producto:** Especifica una descripción más detallada del producto seleccionado. **Unidad de medida:** Especifica en qué unidad de medida se mide el producto seleccionado. **Cantidad:** Especifica la cantidad del producto seleccionado que se va a producir.

Cuadro 3.10: Detalles del requisito de datos RU_04.

Identificador	RD_05
Nombre:	Entrada general
Tipo:	Módulo
Descripción:	Requisito que representa los datos necesarios para la creación de una entrada general.
Casos de uso:	CU_04, CU_05
Datos específicos:	Nº, Nº línea, Tipo, Nº producto, Descripción de producto, Localización, Unidad de medida, Cantidad, Nº mov., Nº lote, Cantidad lote.
Comentarios:	**Nº:** Se especificará el número de serie de la orden de producción a la cual pertenece la entrada general. **Nº línea:** Especifica el número que identifica la entrada general. **Tipo:** Especifica el tipo de entrada general, puede ser de tipo producto o recurso. **Nº producto:** Se seleccionará un producto de la lista de productos disponibles en el sistema. **Descripción de producto:** Especifica una descripción más detallada del producto seleccionado. **Localización:** Se seleccionará una localización disponible en el sistema, la cual indicara en qué lugar está almacenado el lote de la entrada general. **Unidad de medida:** Especifica en qué unidad de medida se mide el producto seleccionado. **Cantidad:** Especifica la cantidad del lote que se quiere utilizar en la entrada general. **Nº mov:** Se seleccionará un movimiento del producto seleccionado. **Nº lote:** Especifica el número de lote que le pertenece al movimiento seleccionado anteriormente. **Cantidad lote:** Especifica la cantidad que contiene el lote en ese momento.

Cuadro 3.11: Detalles del requisito de datos RU_05.

3.2. Análisis de requisitos

En este apartado se realiza un análisis de todos los requisitos del sistema anteriormente nombrados, de esta forma se puede realizar una solución que los cumpla todos de forma correcta. Para realizar dicho análisis se ha optado por la creación de un diagrama de clases.

3.2.1. Diagrama de clases

Un diagrama de clases es una herramienta que permite representar todas las clases utilizadas en el proyecto junto con las asociaciones que tienen, así como el comportamiento de los diferentes objetos y sus relaciones con los otros objetos del sistema.

En la Figura 3.2 se pueden observar las diferentes clases del sistema. En este caso los atributos y métodos de cada una de ellas no se muestran, esto se debe a que más adelante se explicara cada clase con más detalle.

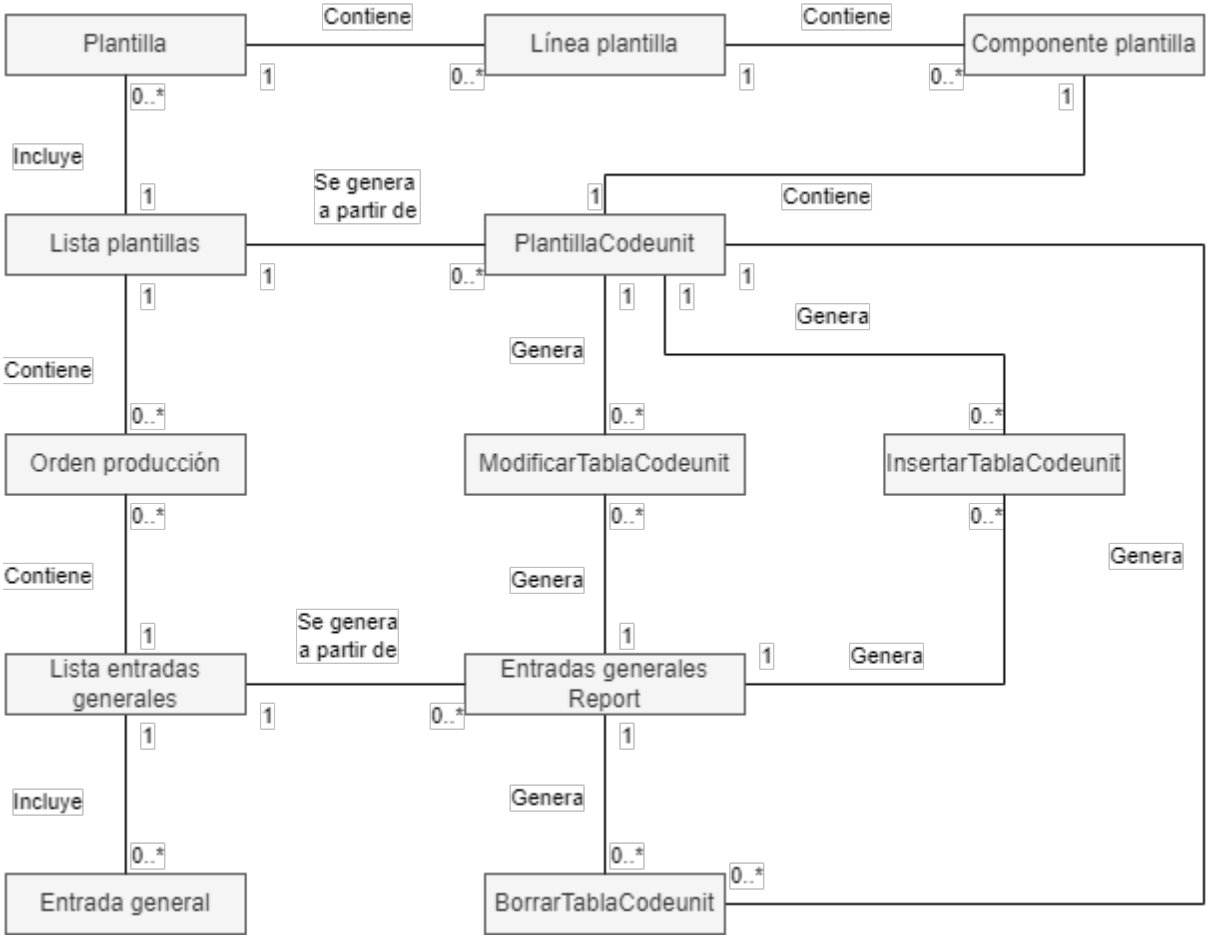

Figura 3.2: Diagrama de clases del proyecto.

Capítulo 4

Diseño del sistema/producto

4.1. Diseño de la arquitectura del sistema/producto

La arquitectura es el modelo que define la estructura, vista y comportamiento de un sistema. De esta manera, es una forma de descripción representativa de cómo funciona un sistema y se comunica con otros componentes del mismo.

En nuestro proyecto, el sistema se compone de tres partes principales, que a su vez se dividen en partes más pequeñas. La primera de esas partes es el cliente web, el cual se encarga de mostrar la interfaz de usuario y enviar las peticiones realizadas al servidor. La segunda parte es el servidor, el cual procesa las peticiones realizadas por la interfaz de usuario y ejecuta las operaciones necesarias, comunicándose con la base de datos. La última parte es la base de datos, en esta se almacenan todos los datos utilizados por la aplicación, esta se comunica con el servidor.

En el caso de BC, que utiliza el formato *SaaS* (Software basado en la nube), el servidor anteriormente nombrado se aloja en la nube. Además del servidor, la base de datos también está en la nube, por lo que no es necesario que el usuario se conecte directamente con la misma.

Todas las conexiones anteriormente nombradas se pueden ver en la Figura 4.1 junto a las partes en las cuales se dividen (interfaz de usuario, servidor y base de datos). Estas partes más pequeñas no se describen en más detalle, esto se debe a que no son de una importancia relevante para el proyecto.

4.2. Diseño de la base de datos

Como se ha mencionado anteriormente, el sistema que se utiliza es BC y debido a que es una herramienta de Microsoft conectado con Azure, se utiliza Azure SQL Database para almacenar los datos.

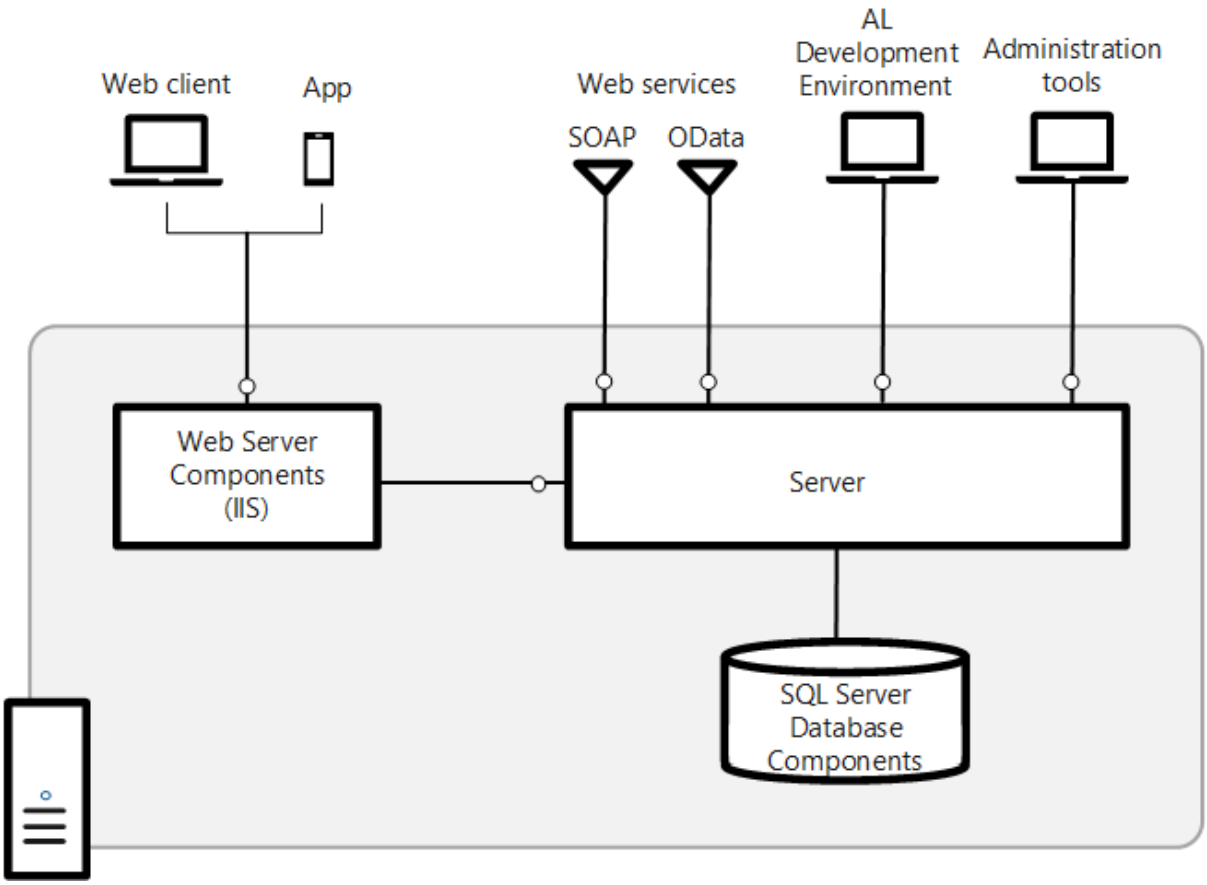

Figura 4.1: Arquitectura de la base de datos en BC [4].

El ERP BC tiene en su programa base una base de datos ya definida, la cual utilizaremos para este proyecto. Además, se han añadido nuevas tablas para la nueva funcionalidad añadida. En la Figura 4.2 se puede observar el diagrama de base de datos final, también veremos que las tablas están separadas en tres tipos diferentes:

- Nuevo: Son tablas que se han creado para almacenar los nuevos datos del sistema.
- Existente: Son tablas que ya existían en el sistema, pero han sido utilizadas en la nueva funcionalidad.
- Extensión: Son cambios que se han realizado en las tablas básicas del sistema.

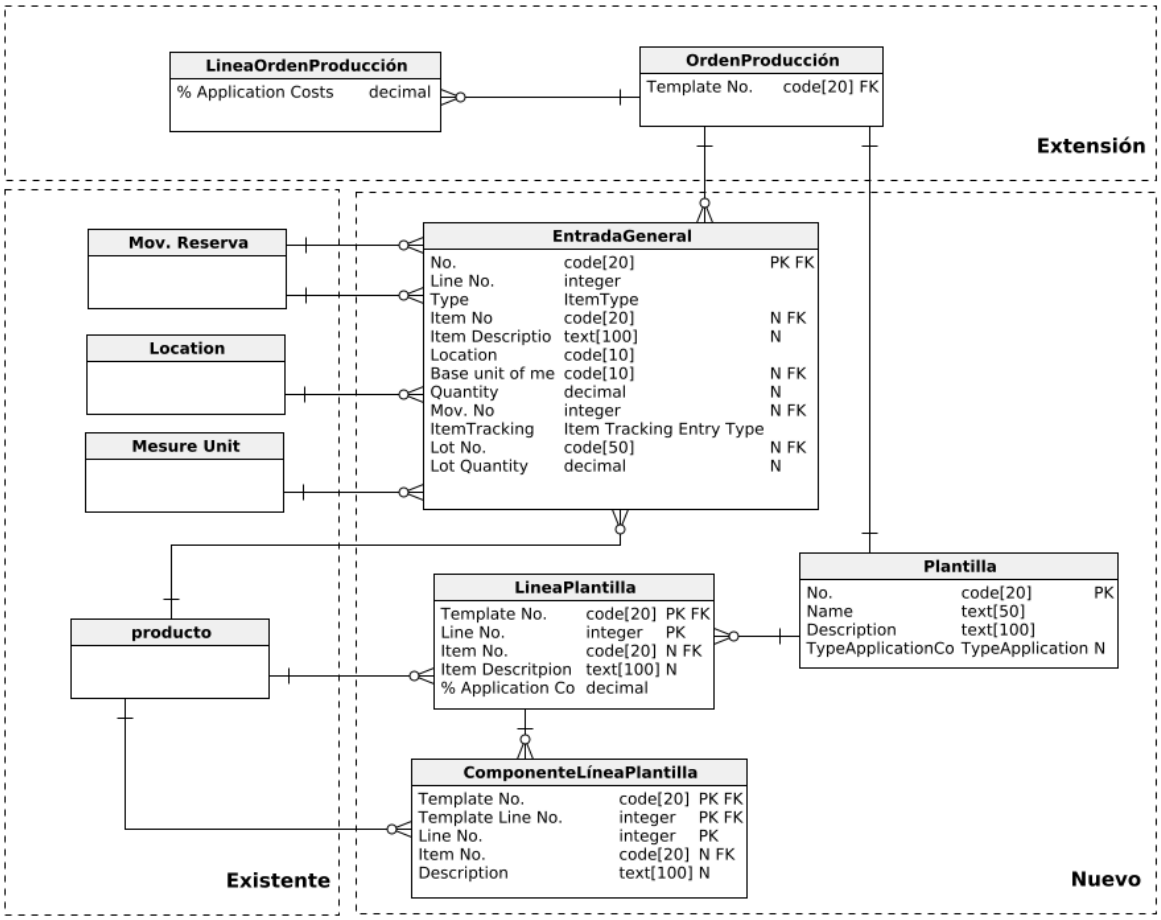

Figura 4.2: Diseño físico de la base de datos del proyecto.

4.3. Diseño de las interfaces

En este apartado se va a mostrar la interfaz creada para el proyecto. En este caso, al estar utilizando un ERP que está desarrollado con una interfaz propia, en el proyecto se ha utilizado la interfaz por defecto del mismo. Esto supone una desventaja, ya que el desarrollador debe adaptarse a un estándar predefinido, pero en contrapunto, el resultado final muestra una interfaz limpia e intuitiva.

Para empezar, se ha añadido una nueva sección en la página de las órdenes de producción planificadas. En esta sección se podrá añadir las diferentes entradas generales que quiera añadir el usuario y generar los nuevos componentes de las líneas de salida. Estos cambios se pueden ver en la Figura 4.3.

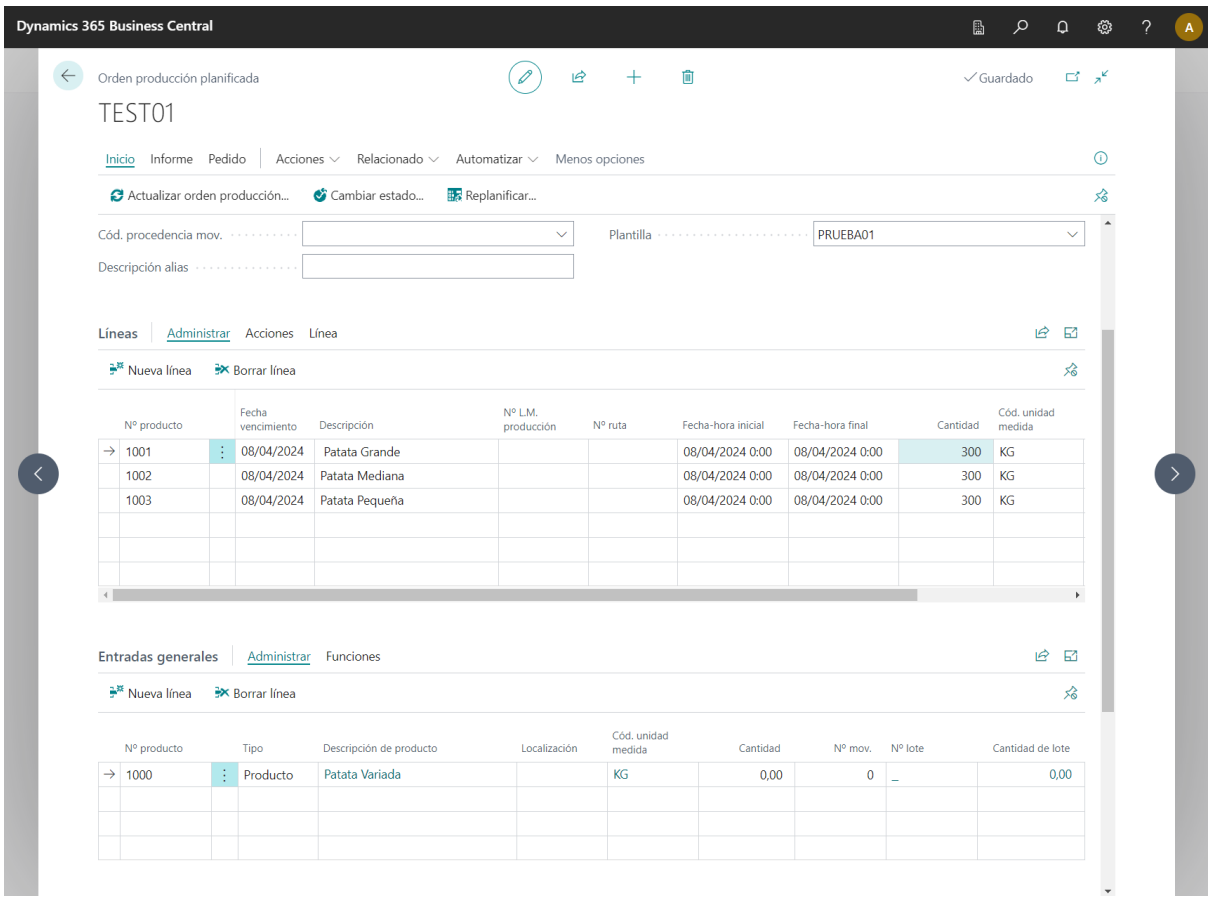

Figura 4.3: Página de orden de producción planificada del producto final.

Además de la funcionalidad básica, se han añadido las plantillas. Para poder visualizarlas se ha creado una lista de plantillas a la cual se puede acceder desde la página de órdenes de producción, como se puede observar en la Figura 4.4. En la Figura 4.5 se puede observar un ejemplo de lista de plantillas.

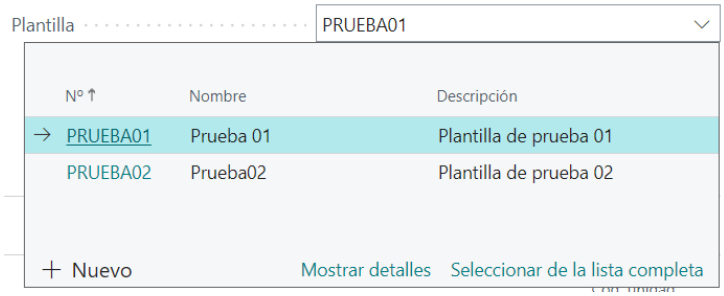

Figura 4.4: Página de acceso a plantillas desde una orden de producción.

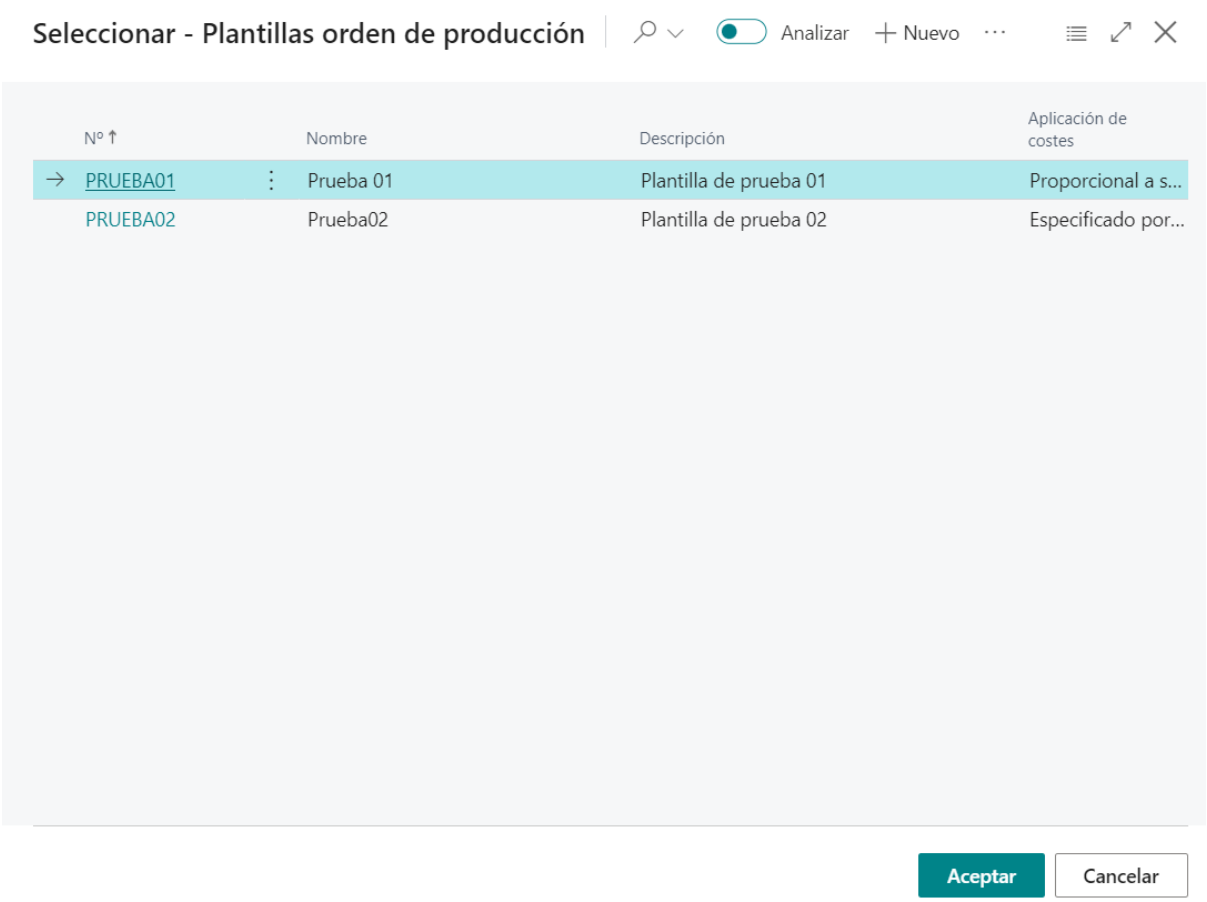

Figura 4.5: Página de lista de plantillas.

Una vez se selecciona una plantilla de la lista, se accede a la página donde se puede modificar dicha plantilla y añadirle líneas de salida que luego pasarán a la orden de producción. En la Figura 4.7 se puede ver un ejemplo de página de plantilla.

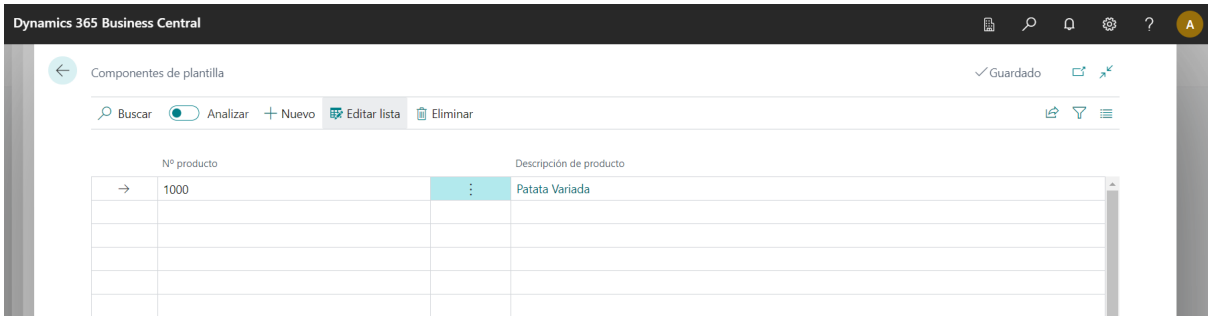

Figura 4.6: Página donde se muestran los componentes de una línea de plantilla.

Finalmente, se puede acceder a una página para ver los componentes que tiene una línea de una plantilla. Esta última página se puede ver en la Figura 4.6.

Como se ha podido observar a lo largo de lo mencionado en el diseño gráfico, BC se centra en hacer listas de elementos a las cuales se puede acceder de diferentes maneras, una vez se selecciona un elemento de la lista se accede a su página, la cual está dividida en diferentes apartados, ofreciendo cada uno de ellos unos datos diferentes.

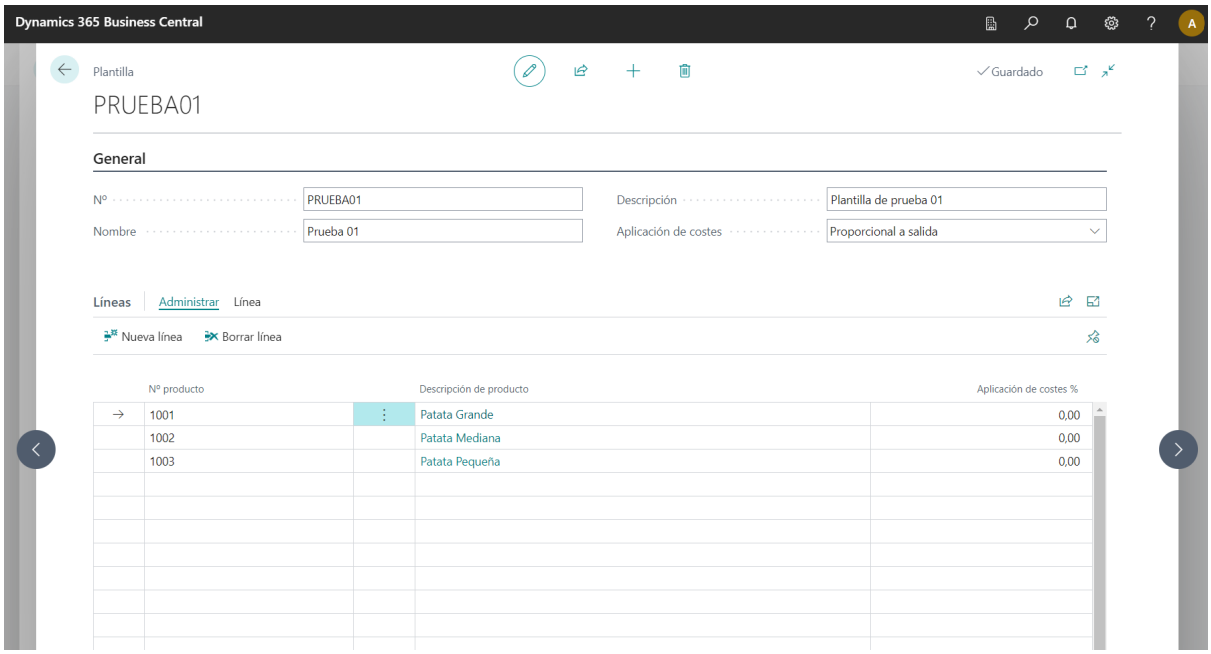

Figura 4.7: Página donde se muestran los datos de una plantilla.

Capítulo 5

Implementación y pruebas

5.1. Estructura del código

En el proyecto se ha seguido la estructura de código utilizada por BC, la cual separa el código en diferentes objetos, los cuales componen las diferentes vistas y datos que se utilizan en BC. En el proyecto se han utilizado varios tipos de objetos, pero los más destacados son los siguientes:

- Table: Objeto que representa una estructura de datos en el sistema. Contiene diferentes atributos que representan los campos de una tabla.
- Tableextension: Objeto que permite modificar tablas que ya existen en el sistema. Permite modificar, añadir y eliminar campos de la tabla original.
- Enum: Objeto que permite crear tipos de datos que se utilizaran en las tablas.
- Page: Objeto que representa la interfaz de un objeto, en esta se muestran los diferentes datos de una tabla asociada. También permite añadir varias funcionalidades adicionales.
- Pageextension: Objeto que permite modificar una página existente. Permite modificar, añadir y eliminar diferentes apartados de la página original.
- Report: Objeto que representa un informe del sistema. Se puede utilizar para generar documentos.
- CodeUnit: Objeto que contiene diferentes funcionalidades, las cuales pueden ser utilizadas por otros objetos del sistema. El resto de los objetos también puede contener funcionalidad concreta, pero se recomienda hacerlo en *"codeunits"* para conseguir un código más ordenado y legible.

El proyecto se ha compuesto de dos directorios principales, el que contiene el código del proyecto y el que contiene las pruebas realizadas para el mismo. Esto se debe a que las pruebas

realizadas se componen de *"codeunits"*, los cuales no pueden pertenecer al directorio donde se encuentre el código del proyecto.

El directorio del proyecto contiene distintos subdirectorios y fichero, los cuales se pueden ver en la Figura 5.1, pero los más relevantes para el proyecto son los siguientes:

- .alpackages: Este subdirectorio contiene el código base que contiene en ese momento BC.
- src: Contiene diferentes subdirectorios en los cuales están divididos los diferentes objetos que componen el proyecto principal.
- Translation: Subdirectorio que contiene los diferentes ficheros .xlf (eXtensible Localization Format) de traducción que contiene el proyecto.
- launch.json: Archivo que contiene toda la configuración de inicio.
- app.json: Archivo que contiene toda la configuración del proyecto.

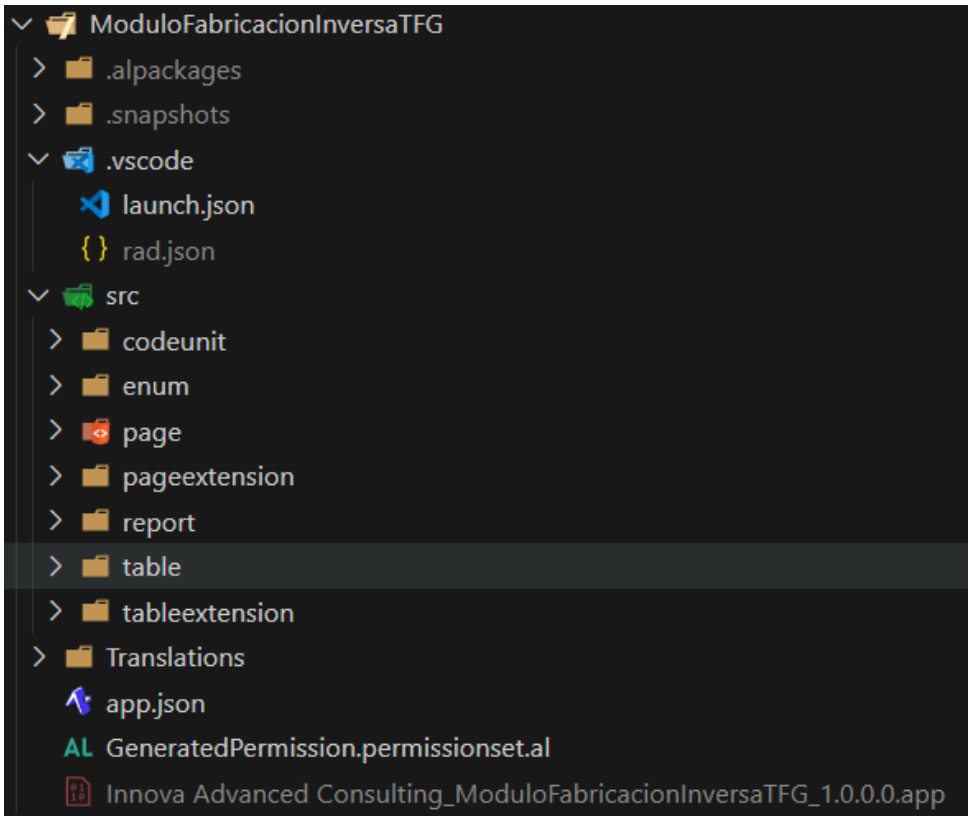

Figura 5.1: Estructura de ficheros y directorios del proyecto.

5.2. Descripción técnica de la implementación

5.2.1. Tablas

Una vez el proyecto está estructurado correctamente, se inicia la creación de las tablas de datos del sistema. Se empieza por crear las estructuras de datos de las entradas generales, seguido de las plantillas.

Para la creación de dichas tablas no se utiliza un sistema habitual de comandos SQL, se utiliza código AL, el cual crea las estructuras de datos en Azure para que puedan ser utilizadas por el entorno en la nube de Business Central. En la Figura 5.2 podemos ver el ejemplo de una tabla, más concretamente de la estructura de datos utilizada para almacenar las entradas generales de una orden de producción.

En la Figura 5.2 también se puede observar que se utiliza en el primer campo de la tabla el atributo *"Comment"*, el cual se utiliza para que los ficheros de traducción, anteriormente nombrados, se generen de forma automática. Esto quiere decir, que una vez se cambie de idioma el ERP Business central, los campos que tengan una traducción se verán en el idioma correcto.

Para que las plantillas y las entradas generales se puedan utilizar en una orden de producción, se modifica la base de datos ya existente en el ERP. Para conseguir esto se crea una *"tableextension"* como se puede ver en la Figura 5.3, de esta manera se pueden añadir nuevos campos o modificar y eliminar los campos ya existentes de una estructura de datos ya creada en BC.

```
tableextension 50100 ProductionOrderTemplate extends "Production Order"
{
    fields
    {
        3 references
        field(50100; Template; Code[20])
        {
            Caption = 'Template', Comment = 'ESP="Plantilla"';
            DataClassification = CustomerContent;
            TableRelation = Template;
        }
    }

    0 references
    trigger OnDelete()
    var
        DeleteTableCU: Codeunit DeleteTable;
    begin
        DeleteTableCU.DeleteGeneralEntry(Rec."No.");
    end;
}
```

Figura 5.3: Ejemplo de una extensión de tabla en BC.

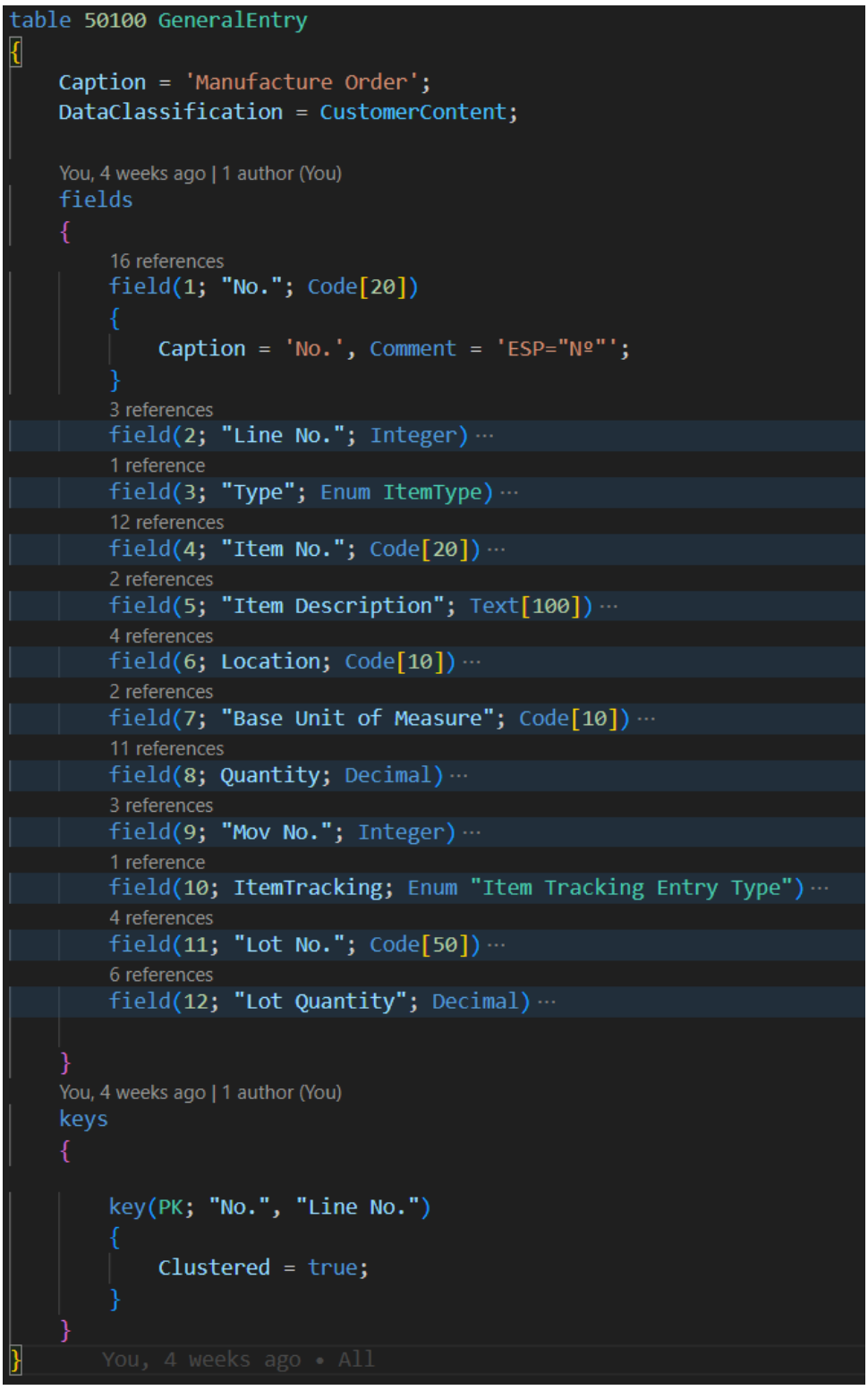

Figura 5.2: Ejemplo de estructura de datos utilizada por BC.

5.2.2. Páginas

Cuando se finaliza la creación y modificación de las estructuras de datos del proyecto, se procede a la creación de las diferentes páginas que confeccionan gran parte de la interfaz gráfica a la que tendrá acceso el usuario. Estas páginas muestran al usuario los datos de las estructuras de datos anteriormente creadas, así como una forma de añadirlos, modificarlos y borrarlos en las estructuras de datos.

Un ejemplo de página, siguiendo la línea del resto de ejemplos, sería la página de entradas generales, la cual se puede ver su código en la Figura 5.4. Esta página es de tipo *"ListPart"* por lo tanto, debe ser añadida a una página de tipo *"Card"* para poder ser vista por el usuario.

```
page 50100 GeneralEntry
{
    Caption = 'General Entrys', Comment = 'ESP="Entradas generales"';
    PageType = ListPart;
    SourceTable = GeneralEntry;
    ApplicationArea = All;
    MultipleNewLines = true;
    DelayedInsert = true;

    You, 4 weeks ago | 1 author (You)
    layout
    {
        You, 4 weeks ago | 1 author (You) | 0 references
        area(content)
        {
            0 references
            repeater(General) ···
        }
    }

    You, 4 weeks ago | 1 author (You)
    actions
    {
        0 references
        area(processing)
        {
            0 references
            group("F&unctions")
            {
                Caption = 'F&unctions', Comment = 'ESP="Funciones"';
                Image = "Action";
                0 references
                action("GenerateComponents") ···
            }
        }
    }
```

Figura 5.4: Ejemplo de una página en BC.

En el caso de la página anteriormente mencionada, se debe añadir a una página ya existente en la base de BC, esto se hace mediante una *"pageextension"*, la cual sirve para añadir, borrar

y modificar páginas ya creadas. Un ejemplo de *"pageextension"* es la que podemos ver en la Figura 5.5. En este proyecto se trabaja sobre una funcionalidad ya existente en BC junto a nuevas funcionalidades que no tenía previamente, de esta manera se crean varios objetos de tipo *"page"* y *"pageextension"* para poder implementar las interfaces de usuario necesarias.

```
pageextension 50100 PlannedGeneralEntry extends "Planned Production Order"
{
    You, 4 weeks ago | 1 author (You)
    layout
    {
        addlast(General)
        {
            0 references
            field(Template; Rec.Template)
            {
                Caption = 'Template', Comment = 'ESP="Plantilla"';
                ApplicationArea = All;
                ToolTip = 'Specifies the template used for the production order.'

                0 references
                trigger OnValidate()
                var
                    TemplateLinesRecord: Record TemplateLines;
                    TemplateCU: Codeunit Template;
                begin
                    TemplateLinesRecord.SetRange("Template No.", Rec.Template);
                    TemplateCU.SetNoProductionOrder(Rec."No.");
                    TemplateCU.Run(TemplateLinesRecord);
                end;
            }
        }
        addafter(ProdOrderLines)        You, 4 weeks ago • All
        {
            You, 4 weeks ago | 1 author (You) | 0 references
            part("ProductionOrders"; GeneralEntry)
            {
                ApplicationArea = Manufacturing;
                SubPageLink = "No." = field("No.");
            }
        }
    }
}
```

Figura 5.5: Ejemplo de una extensión de tabla en BC.

5.2.3. Reports

Una vez la interfaz de usuario y las estructuras de datos del proyecto están finalizadas, se añade la funcionalidad necesaria para cumplir los requisitos del cliente. En este proyecto la estructura principal del proyecto es un objeto de tipo *"Record"*, estos objetos sirven comúnmente para generar documentos con unos campos predefinidos, los cuales el usuario podrá filtrar y modificar antes de su creación.

En este caso concreto, se va a utilizar una variante de *"report"* la cual tiene el atributo de *"ProcessingOnly"*, este atributo hace que no se genere un documento, sino que recorre la información de la tabla establecida en el *"report"* fijando una línea de la tabla cada vez. De esta forma se genera un bucle que recorre todo el contenido de la tabla de una forma sencilla.

En la Figura 5.6 podemos ver el *"report"* que se utiliza en este proyecto, más concretamente las tablas a las cuales accede (Production Order, Template y Prod. Order Line) mediante la etiqueta *"dataitem"*. Dentro de estas etiquetas se accede a *"codeunits"*, los cuales se explican más adelante, que contienen el código que se va a ejecutar para cada línea de la tabla.

En el *"report"* también se relacionan entre sí las diferentes tablas que se utilizan y se establecen diferentes filtros para poder acceder más concretamente a los datos que se van a utilizar, de esta manera se puede simplificar el tiempo de acceso a la información mejorando de forma significativa el tiempo de ejecución del programa.

Una vez explicados los terminas generales de un *"report"*, el de nuestro proyecto busca las entradas generales que ha creado el usuario e inserta en las líneas de salida, de la orden de producción actual, todos los componentes que el usuario ha introducido como entradas generales. Se pueden diferenciar dos formas de hacerlo, una que solo modifica los componentes que ya existen en las líneas de salida y otro que genera componentes nuevos.

5.2.4. CodeUnits

Los *"codeunits"* son los objetos más funcionales del proyecto, estos contienen la gran mayoría de métodos que van a ser utilizados por el resto de los objetos del programa. En este caso, se han creado principalmente tres *"codeunits"*, los cuales separan las funcionalidades de inserción, modificación y eliminación de entradas en las diferentes tablas de datos. Su funcionamiento es sencillo, reciben una entrada concreta de una tabla y hacen las modificaciones necesarias. En las Figuras 5.7, 5.9, 5.8 podemos ver un método de inserción, modificación y eliminación respectivamente.

En el proyecto estos métodos se encargan de hacer los cambios necesarios en la base de datos para que la información de BC se mantenga actualizada en todo momento, no solo la información a la que el usuario tiene acceso. También hay otros *"codeunits"* que se utilizan para calcular las cantidades de cada componente, así como actualizar la información de una orden de producción cuando se cambia la plantilla de la misma.

```
report 50100 GeneralEntry
{
    Caption = 'General Entry', Comment = 'ESP="Entrada general"';
    ProcessingOnly = true;
    UseRequestPage = false;

    You, 4 weeks ago | 1 author (You)
    dataset
    {

        4 references
        dataitem(ProductionOrder; "Production Order")
        {
            1 reference
            dataitem(Template; Template)
            {
                DataItemLinkReference = ProductionOrder;
                DataItemLink = "No." = field(Template);
            }

            3 references
            dataitem(ProdOrderLine; "Prod. Order Line")
            {
                DataItemLinkReference = ProductionOrder;
                DataItemLink = "Prod. Order No." = field("No.");
                3 references
                dataitem(GeneralEntry; GeneralEntry)
                {
                    DataItemLinkReference = ProdOrderLine;
                    DataItemLink = "No." = field("Prod. Order No.");

                    0 references
                    trigger OnAfterGetRecord()
                    var
                        QuantityItem: Decimal;
                    begin
                        QuantityItem := UtilityCU.CalcSumQuantityItem(NoCode, GeneralEntry."Item No.");

                        if ("No." = '') then
                            CurrReport.Break()
                        else
                            if Template.TypeApplicationCosts = TypeApplication.FromInteger(1) then
                                InsertTableCU.InsertComponentLineSpecified(ProdOrderLine, GeneralEntry)
                            else
                                ModifyTableCU.ModifyComponent(ProdOrderLine, GeneralEntry, QuantityItem)
                    end;
                }
            }
```

Figura 5.6: Ejemplo de un *report* en BC.

Como podemos ver en la Figura 5.7, se utilizan diferentes líneas a la hora de modificar una base de datos mediante código, pero principalmente podemos ver que se utiliza *"Init"* para iniciar una nueva entrada en una tabla, *"Validate"* para que se apliquen las restricciones que se han creado en la base de datos y se calculen los campos que hacen referencia a otras tablas y *"CalcFields"* que se encargan de generar datos para poder guardarlos en la base de datos.

```
1 reference
procedure InsertReservationEntry(GeneralEntry: Record GeneralEntry; ProdOrderLine: Integer; LineNo: Integer;
var
    ReservationEntryRecord: Record "Reservation Entry";
begin
    ReservationEntryRecord.Init();
    ReservationEntryRecord."Entry No." := 0;
    ReservationEntryRecord.Validate("Item No.", GeneralEntry."Item No.");
    ReservationEntryRecord."Location Code" := GeneralEntry.Location;
    ReservationEntryRecord.Validate("Quantity (Base)", -InsertQuantity);
    ReservationEntryRecord."Reservation Status" := "Reservation Status"::Prospect;
    ReservationEntryRecord."Creation Date" := WorkDate();
    ReservationEntryRecord."Source Type" := 5407;
    ReservationEntryRecord."Source Subtype" := 1;
    ReservationEntryRecord."Source ID" := GeneralEntry."No.";
    ReservationEntryRecord."Source Prod. Order Line" := ProdOrderLine;
    ReservationEntryRecord."Source Ref. No." := LineNo;
    ReservationEntryRecord."Shipment Date" := WorkDate();
    ReservationEntryRecord."Item Tracking" := "Item Tracking Entry Type".FromInteger(1);
    GeneralEntry.CalcFields("Lot No.");
    ReservationEntryRecord.Validate("Lot No.", GeneralEntry."Lot No.");

    ReservationEntryRecord.Insert(true);
end;
```

Figura 5.7: Ejemplo de un método de inserción.

```
codeunit 50101 DeleteTable
{
    2 references
    procedure DeleteGeneralEntry(NoProdOrder: Code[20])
    var
        GeneralEntry: Record GeneralEntry;
    begin
        GeneralEntry.SetRange("No.", NoProdOrder);
        GeneralEntry.DeleteAll(true);
    end;
```

Figura 5.8: Ejemplo de un método de eliminación.

```
codeunit 50102 ModifyTable
    procedure ModifyComponent(ProdOrderLine: Record "Prod. Order Line"; GeneralEntry: Record Gene
    var
        ProdOrderComponentRecord: Record "Prod. Order Component";
        LineNo: Integer;
    begin
        ProdOrderComponentRecord.SetRange("Prod. Order No.", GeneralEntry."No.");
        ProdOrderComponentRecord.SetRange("Prod. Order Line No.", ProdOrderLine."Line No.");
        ProdOrderComponentRecord.SetRange("Item No.", GeneralEntry."Item No.");
        if ProdOrderComponentRecord.FindFirst() then begin
            LineNo := ProdOrderComponentRecord."Line No.";

            ProdOrderComponentRecord.Init();
            ProdOrderComponentRecord.Validate(Status, "Production Order Status"::Planned);
            ProdOrderComponentRecord."Prod. Order No." := GeneralEntry."No.";
            ProdOrderComponentRecord."Prod. Order Line No." := ProdOrderLine."Line No.";
            ProdOrderComponentRecord."Line No." := LineNo;
            ProdOrderComponentRecord.Validate("Item No.", GeneralEntry."Item No.");
            ProdOrderComponentRecord.Validate("Location Code", GeneralEntry.Location);
            ProdOrderComponentRecord.Validate("Expected Qty. (Base)", (ProdOrderLine.Quantity * G
            ProdOrderComponentRecord.Validate("Quantity per", ProdOrderComponentRecord."Expected

            ProdOrderComponentRecord.Validate("Unit of Measure Code");
            ProdOrderComponentRecord.Validate("Routing Link Code");
            ProdOrderComponentRecord.Validate("Scrap %");
            ProdOrderComponentRecord.Validate("Variant Code");
            ProdOrderComponentRecord.Validate("Flushing Method");
            ProdOrderComponentRecord.Validate("Location Code");
            ProdOrderComponentRecord.Validate("Shortcut Dimension 1 Code");
            ProdOrderComponentRecord.Validate("Shortcut Dimension 2 Code");
            ProdOrderComponentRecord.Validate("Bin Code");
            ProdOrderComponentRecord.Validate("Length");
            ProdOrderComponentRecord.Validate(Width);
            ProdOrderComponentRecord.Validate(Depth);
            ProdOrderComponentRecord.Validate("Calculation Formula");
            ProdOrderComponentRecord.Validate("Unit Cost");      You, last month • All
            ProdOrderComponentRecord.Validate("Due Date");
            ProdOrderComponentRecord.Validate("Due Time");
            ProdOrderComponentRecord.Validate("Expected Qty. (Base)");
            ProdOrderComponentRecord.Validate("Due Date-Time");
            ProdOrderComponentRecord.Validate("Dimension Set ID");
            ProdOrderComponentRecord.Validate("Qty. Picked");
            ProdOrderComponentRecord.Validate("Pick Qty. (Base)");
            ProdOrderComponentRecord.Validate("Indirect Cost %");
            ProdOrderComponentRecord.Validate("Overhead Rate");

            ProdOrderComponentRecord.Modify();
```

Figura 5.9: Ejemplo de un método de modificación.

5.3. Verificación y validación

En este apartado se explica cómo se han realizado las pruebas en el entorno de BC. En el caso de BC hay que tener en cuenta que es un ERP expuesto a constantes actualizaciones, por lo tanto, hay que realizar pruebas para comprobar el correcto funcionamiento de los módulos con el cambio de versiones.

En el caso de este proyecto se ha creado un nuevo repositorio, en el cual, se ha creado un *"codeunit"* para cada prueba que se quiere realizar. Estas pruebas siguen un patrón Given-When-Then [6], el cual consiste en dividir cada prueba en tres partes diferentes:

- Given: Especifica los valores con los que se inicia la prueba.
- When: Especifica qué acciones se van a tomar en la prueba.
- Then: Especifica el resultado que se espera en la prueba.

En la Figura 5.10 se puede observar la prueba creada para comprobar si la plantilla se aplica de forma correcta a una orden de producción, en esta se puede ver como la prueba está dividida en las tres partes que se han nombrado anteriormente.

Por último, se ha decidido añadir todas las pruebas a un entorno de integración continua. Como se ha explicado en las tecnologías utilizadas, el proyecto se ha guardado en un repositorio de GitHub, dicho repositorio se ha creado mediante el repositorio AL-GO [1]. Este repositorio contiene un conjunto de plantillas que permiten que las pruebas creadas en BC se ejecuten en un entorno simulado cada vez que se añade una nueva versión del proyecto al repositorio. En la Figura 5.11 podemos ver un ejemplo de esta integración continua.

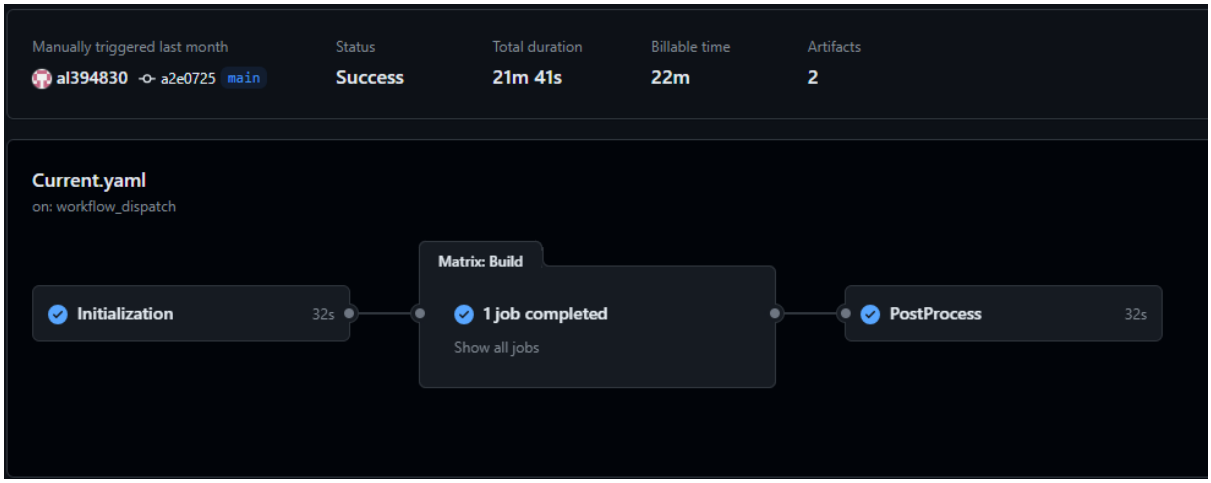

Figura 5.11: Ejemplo de integración continua en GIT.

```
codeunit 50151 TemplateTest
{
    Subtype = Test;

    [Test]
    0 references
    procedure AddTemplate()
    var
        ItemUnitOfMesureRecord: Record "Item unit of Measure";
        UnitOfMesureRecord: Record "Unit of Measure";
        ItemRecord: Record Item;
        TemplateRecord: Record Template;
        TemplateLineRecord: Record TemplateLines;
        TemplateComponentRecord: Record TemplateLineComponent;
        ProductionOrderRecord: Record "Production Order";
        ProdOrderLineRecord: Record "Prod. Order Line";
        ProdOrderComponentRecord: Record "Prod. Order Component";

        TemplateCU: Codeunit Template;

    begin
        //[Given]
        CreateNewUnitOfMesure(UnitOfMesureRecord);
        CreateNewItemRecord(ItemRecord);
        CreateNewItemUnitOfMesure(ItemUnitOfMesureRecord);
        CreateNewTemplateRecord(TemplateRecord, TypeApplication::"Line specified");
        CreateNewTemplateLineRecord(TemplateLineRecord);
        CreateNewTemplateComponentRecord(TemplateComponentRecord);
        CreateNewProductionOrderRecord(ProductionOrderRecord, true);

        //[When]
        TemplateLineRecord.SetRange("Template No.", 'TEST');
        TemplateCU.SetNoProductionOrder('TEST');
        TemplateCU.Run(TemplateLineRecord);

        //[Then]
        ProdOrderLineRecord.SetRange("Prod. Order No.", 'TEST');
        ProdOrderLineRecord.SetRange("Line No.", 10000);

        ProdOrderComponentRecord.SetRange("Prod. Order No.", 'TEST');
        ProdOrderComponentRecord.SetRange("Prod. Order Line No.", 10000);
        ProdOrderComponentRecord.SetRange("Line No.", 10000);

        Assert.AreEqual(1, ProdOrderLineRecord.Count(), 'ERROR ProdOrderLineRecord');
        Assert.AreEqual(1, ProdOrderComponentRecord.Count(), 'ERROR ProdOrderComponentRecord');

    end;
```

Figura 5.10: Ejemplo de una prueba creada en AL.

Capítulo 6

Conclusiones

En este proyecto, a nivel técnico, se ha conseguido completar todas las funciones que inicialmente se esperaban del mismo. En cuanto a nivel de usuario, se esperaba poder completar un manual de usuario para facilitar su futuro uso, pero finalmente se decidió añadir funcionalidad adicional como las plantillas, ya que aportan una mayor comodidad a la hora de realizar tareas repetitivas.

A nivel profesional, realizar este proyecto en una empresa me ha permitido ver cómo se trabaja y cómo funcionan realmente. Esta empresa concretamente me ha permitido iniciarme en el mundo de los ERP, aprendiendo un nuevo lenguaje, el cual se utiliza en el ERP BC.

En el ámbito personal, este proyecto me ha permitido aprender a programar en algo completamente nuevo, esto me ha obligado a abandonar mi zona de confort en un lugar completamente nuevo, aprender y conocer a un equipo, al cual he tenido que adaptar mi forma de trabajar. Finalmente, decir que este proyecto es el más importante que he realizado hasta el momento, no solo en dificultad técnica, ya que simboliza el fin de la universidad y el inicio de mi carrera laboral.

Bibliografía

[1] Freddy Kristiansen. AL-Go para GitHub. https://github.com/microsoft/AL-Go. [Consulta: 4 de Mayo de 2024].

[2] Lucidchart.com. Creación de diagramas. https://www.lucidchart.com/pages/examples/uml_diagram_tool.

[3] Luis Asdrubal Carmona. Qué es un método predictivo y cuándo usarlo. https://modoproyecto.com/que-es-un-metodo-predictivo-y-cuando-usarlo/. [Consulta: 3 de Mayo de 2024].

[4] Microsoft.com. Arquitectura de la base de datos en bc. https://learn.microsoft.com/en-us/dynamics365/business-central/dev-itpro/deployment/product-and-architecture-overview.

[5] Microsoft.com. Dynamics 365 business central. https://www.microsoft.com/es-es/dynamics-365/products/business-central. [Consulta: 24 de Abril de 2024].

[6] Kenana Rhoton. Given-when-then. https://www.redsauce.net/blog/es/3-parts-software-test. [Consulta: 2 de Mayo de 2024].

[7] Talent.com. Salarios para trabajadores en business central. https://es.talent.com/.