Grado en Ingeniería Informática

Trabajo de Final de Grado

Registro de gastos en Business Central a través de Power Apps.

Autor:
Pau Castejón Serrano

Supervisor:
Sergi Vilar Domènech
Tutor académico:
Begoña Martínez Salvador

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

Resumen

El presente documento presenta la memoria del trabajo de final de grado realizada por el alumno Pau Castejón Serrano durante la estancia en prácticas en la empresa Innova Advanced Consulting S.L. Esta memoria describe el desarrollo de un proyecto cuyo objetivo es la creación de una aplicación móvil en Power Apps, que facilite la imputación de los gastos corrientes de una empresa en el ERP de Microsoft, Business Central.

Para el desarrollo del proyecto se han utilizado dos herramientas de Microsoft con el objetivo de que las conexiones entre ellas y el ERP sean mucho más fáciles. Para la creación de la app se ha utilizado Power Apps y para realizar la conexión con Business Central se han utilizado flujos de Power Automate.

La aplicación desarrollada permite a un usuario consultar sus gastos, registrar nuevos gastos, posibilitando adjuntar una foto del ticket o factura. Los nuevos gastos introducidos se registran en Business Central a la espera que un responsable como pueda ser, el contable de la empresa, revise el gasto realizado, introduzca la información sensible restante y registre el gasto.

Palabras clave

ERP, Dynamics 365 Business Central, Microsoft Power Apps, gastos corrientes, adjuntar factura

Keywords

ERP, Dynamics 365 Business Central, Microsoft Power Apps, current expenses, attaching invoice

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

1.1. Contexto y motivación del proyecto

El proyecto a desarrollar es una propuesta de la empresa Innova Advanced Consulting SL, situada en Castellón, la cual se especializa en el mundo de los ERP de Microsoft. La empresa cuenta con mas de 19 años en el sector y está formada por una plantilla de mas de 40 empleados. Además tienen la confianza de más de 300 partners de Microsoft.

La empresa se dedica a desarrollar aplicaciones, módulos o versiones personalizadas del ERP de Microsoft Dynamics 365 Business Central(BC) y su posterior soporte. Principalmente estos servicios son contratados a través de otros asociados que son quienes están en contacto con el cliente final. Recientemente Innova está comportándose como partner de Microsoft actuando así directamente con el cliente final.

La principal motivación del proyecto es mejorar la eficiencia y accesibilidad en la imputación de gastos de los trabajadores en Microsoft Dynamics 365 Business Central mediante una app desarrollada en Power Apps. Actualmente, para registrar los gastos, el empleado debe enviar todas las facturas al contable o esperar a reunirse con éste para dárselas físicamente. La implementación de una interfaz móvil con capacidad de adjuntar imágenes facilitará la entrada de datos y agilizará el proceso, mejorando tanto la experiencia del usuario como la precisión de la información registrada en el ERP.

1.2. Objetivo y alcance del producto

El objetivo del producto es el desarrollo de una app para que el propio trabajador pueda registrar gastos corrientes, como dietas. En lugar de esperar a que el trabajador regrese para que el contable los ingrese, el propio trabajador podrá ingresar estos datos desde la aplicación, esté donde esté, reduciendo así el número de intermediarios en la introducción de gastos y por tanto, de posibles errores.

1.2.1. Alcance funcional

Las funcionalidades generales del producto que se van a desarrollar son las siguientes:

- El usuario podrá imputar gastos
- Consulta de los gastos imputados anteriormente
- Podrá adjuntar imágenes de documentos relacionados con los gastos y así respaldar la transacción
- Podrá descargar los documentos relacionados con los gastos
- Inserción automática de los gastos en Microsoft Dynamics 365 Business Central en el proyecto y tarea que corresponda.

1.2.2. Alcance organizativo

En lo que se refiere al alcance organizativo, toda empresa que haga uso o sea gestionada mediante Business Central y adquiera la aplicación desarrollada podría utilizarla para beneficiarse de la imputación de gastos corrientes.

1.2.3. Alcance tecnológico

Desde el punto de vista tecnológico la aplicación debe poder conectarse con Micrososft Dynamics 365 Business Central mediante un flujo de datos de Microsoft Power Automate para así poder hacer una imputación de los gastos corrientes y que se vea reflejado en los informes de contabilidad.

1.3. Objetivo y alcance del proyecto

El principal objetivo de este proyecto es el desarrollo de una aplicación móvil de uso general, compatible con dispositivos Android e iOS, que se conecte con el ERP de Microsoft para facilitar y agilizar el proceso de imputación de gastos corrientes.

El objetivo principal se puede desglosar en los siguientes subobjetivos:

- Investigar y estudiar Microsoft Dynamics 365 Business Central.
- Aprender el lenguaje Application Language(AL) que se utiliza para programar en Business Central.
- Estudiar los requisitos de la aplicación demandados por el cliente.

- Desarrollar un nuevo módulo dentro de BC para personalizar el ERP de Microsoft.
- Desarrollar una aplicación móvil con Power Apps.
- Desarrollar un flujo de datos en Power Automate.
- Permitir que los trabajadores puedan imputar gastos desde la aplicación móvil.
- Permitir la adjunción de la foto de una factura en el gasto desde la aplicación móvil.
- Permitir el registro de datos en el proyecto y tarea que correspondan.
- Realización de pruebas e integración en un pipeline de CI/CD en GitHub.
- Generar una presentación de la nueva funcionalidad.

1.4. Descripción detallada del proyecto

En la actualidad, BC, es un ERP ampliamente utilizado por pequeñas y medianas empresas para gestionar los procesos empresariales. BC incluye funcionalidades para muchas áreas, como pueden ser los módulos de ventas, compra, inventario, fabricación o proyectos.

No obstante, hay funcionalidades que no existen o módulos que podrían mejorarse. Para esto se desarrollan extensiones, que se adaptan a las necesidades del cliente. En este caso, la facilitación del registro de gastos corrientes.

Actualmente la gestión de gastos corrientes en muchos casos no se realiza en Business Central, ya que no hay ningún módulo especifico. En otros casos, se insertan desde el Diario general de proyectos, el cual se encarga de almacenar lineas que corresponden a transacciones contables en general y registrar estas lineas en la tarea y proyecto que se indican. El problema reside en que no hay ninguna manera de identificar que es un gasto corriente y que no. Además de que si queremos adjuntar un documento tenemos que realizar la foto de este, pasarla al ordenador y adjuntar el archivo.

Para solucionar este problema se ha desarrollado una aplicación móvil que permitirá la consulta de los gastos insertados anteriormente y la inserción de estos gastos desde cualquier lugar, permitiendo adjuntar una imagen de la factura. El proceso de inserción registrará una linea en el Diario general de proyectos, ahorrando al usuario tener que insertar esta linea, rellenando cada campo. Antes de la inserción de los gastos en el proyecto y tarea que les corresponda, en BC un responsable como pueda ser, el contable o el administrador empresarial, rellenará los datos sensibles, los cuales tiene que ver con las cuentas bancarias. Por último, también se permite la visualización en BC de todos los gastos insertados permitiendo así descargar por si es necesario la factura o ticket.

1.4.1. Tecnologías

Las tecnologías que se van a utilizar durante el desarrollo del proyecto son las siguientes:

Dynamics 365 Business Central: diseñado especialmente para empresas de tamaño pequeño y mediano, Microsoft Dynamics 365 Business Central representa una solución innovadora en el ámbito del software de gestión empresarial. Este sistema integra todas las funciones en un solo lugar, ofreciendo un completo ERP que abarca áreas como finanzas, manufactura, ventas, logística, gestión de proyectos y atención al cliente. En este proyecto se desarrollará una extensión que cree las estructuras y acciones necesarias para la imputación de gastos.

Microsoft Power Apps: esta herramienta facilita la creación de aplicaciones personalizadas con mínima programación, lo que permite automatizar tareas cotidianas y ayudar a la realización de procesos empresariales. En este proyecto se utilizará Power Apps para la realización de una aplicación que permita la consulta de los gastos corrientes e insertar nuevos gastos.

Microsoft Power Automate: es un servicio en la nube que simplifica la creación de flujos de trabajo para usuarios de negocios, permitiéndoles automatizar tareas y procesos que suelen ser laboriosos, utilizando diversas aplicaciones y servicios de manera práctica y accesible. En este proyecto se creará un flujo de trabajo que facilité la conexión entre BC y Power Apps, para insertar los datos directamente en el ERP. Tanto Power Apps como Power Automate forman parte de Power Platform [5], una suite de Microsoft que ofrece distintos programas que ayudan a la automatización de procesos empresariales

Visual Studio Code: es un editor de código fuente el cual utilizaremos para el desarrollo de extensiones en BC utilizando el lenguaje AL. Proporciona una amplia gama de extensiones que ayudaran al desarrollo, la depuración y la gestión de proyectos en BC.

Github: se trata de una plataforma en línea para la gestión del desarrollo de software, utilizada para almacenar, supervisar y colaborar en proyectos de software. Permite compartir archivos de código y trabajar en proyectos de código abierto de manera colaborativa, realizar seguimiento del código, la gestión de varias versiones del proyecto y realizar revisiones de código. Este proyecto se alojará en Github a partir de una plantilla llamada AL-Go, creada por Microsoft. La plantilla tiene configurada varios flujos desarrollados en Github Actions, un servicio de integración y entrega continua que Github proporciona. Estos flujos permitirán que se pruebe constantemente nuestro código al ser modificado, ejecutando un conjunto de pruebas creado anteriormente.

AL: como lenguaje en este proyecto se va a utilizar el AL(Application Language), un lenguaje de programación utilizado para escribir código en BC, permitiendo así poder crear objetos como páginas, tablas o leer, escribir y cambiar datos en BC.

1.5. Estructura de la memoria

La organización de lo que resta de la memoria será la siguiente:

- **Planificación del proyecto:** se especifica la metodología de desarrollo del proyecto y según esta se mostrará una planificación inicial seguida de su seguimiento y como acaba. Además, se explican los costes y los posibles riesgos del desarrollo del proyecto.

- **Análisis del sistema/producto:** en este apartado se indicarán los requisitos con los que deberá contar el producto final. También un análisis de los requisitos donde se mostrará como el producto alcanzará los requisitos indicados anteriormente.
- **Diseño del sistema/producto:** aquí se muestra los distintos diseños del producto: diseño de la base de datos, diseño de la arquitectura del producto y por último, diseño de las interfaces.
- **Implementación y pruebas**: en este apartado se muestran, de manera general, la estructura del código desarrollado y como funciona, así como otros aspectos relacionados con el desarrollo del producto. Por último, se mostrarán las pruebas creadas que ayudan a la verificación y validación del producto.
- **Conclusiones**: finalmente, se comentarán los resultados alcanzados teniendo en cuenta los objetivos planteados inicialmente. Además, se incluirá una reflexión sobre los conocimientos adquiridos tanto en el ámbito formativo como profesional.

Capítulo 2

Planificación del proyecto

2.1. Metodología

Para este proyecto se ha considerado la utilización de una metodología predictiva ya que el proyecto parte de una especificación inicial la cual se prevé que no variará. Esta metodología es útil en entornos donde los requisitos son estables, los riesgos son conocidos y existen plazos definidos.

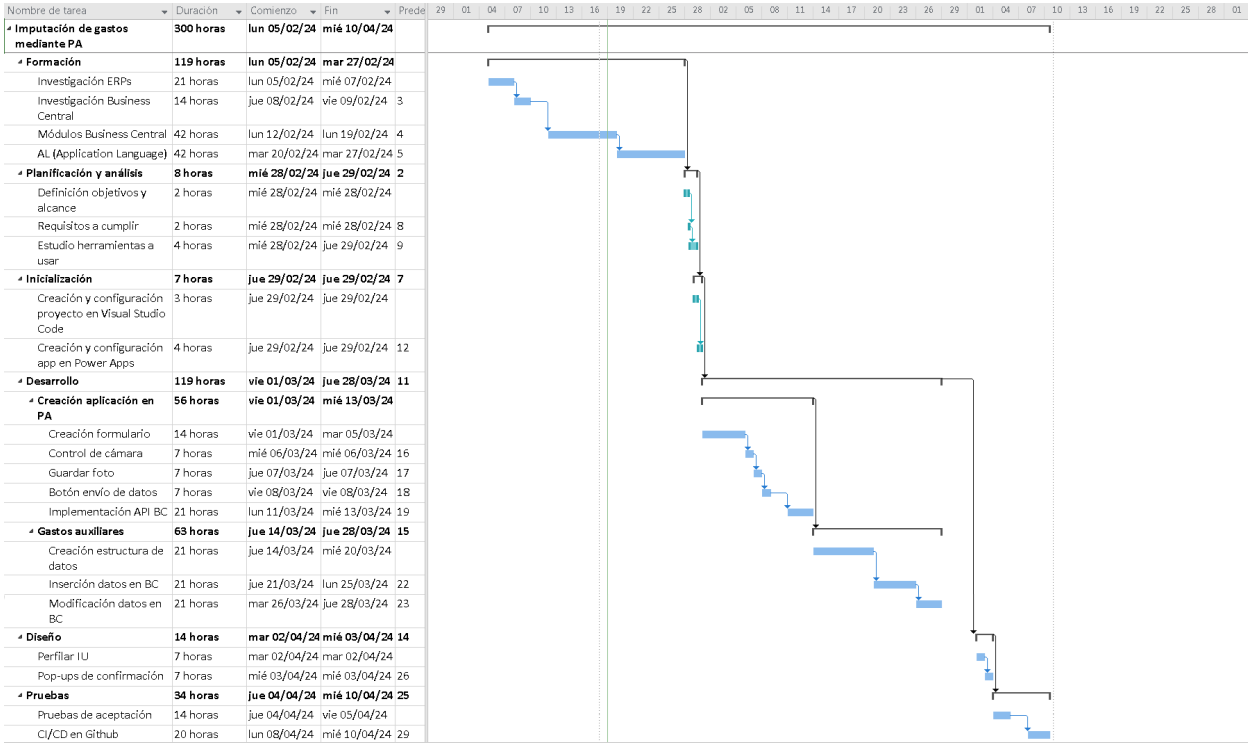
Las actividades o tareas que componen el proyecto son las siguientes:

- Formación: al inicio del proyecto se realiza una formación para familiarizarse con las tecnologías que se usarán.
- Planificación y análisis: se realiza un estudio de los requisitos para definir qué objetivos se deberán realizar y cumplir. A su vez, se realiza un estudio sobre como se utilizarán las herramientas para completar los objetivos.
- Inicialización: donde se establecen las bases, configurando todas las herramientas que se usarán para poder desarrollar la solución.
- Desarrollo: comienza la implementación de la solución.
- Diseño: una vez acabado el desarrollo de la solución, se modifican las interfaces necesarias para ser amigables para el usuario.
- Pruebas y depuración: la realización de pruebas que compruebe el correcto funcionamiento de la aplicación.

2.2. Planificación temporal del proyecto

Para la estimación del tiempo de las tareas y actividades que componen el proyecto se ha realizado el diagrama de la figura 2.1, que incluye en más detalle las fases anteriormente

descritas.



Nombre de tarea	Duración	Comienzo	Fin	Prede
Imputación de gastos mediante PA	**300 horas**	**lun 05/02/24**	**mié 10/04/24**	
Formación	**119 horas**	**lun 05/02/24**	**mar 27/02/24**	
Investigación ERPs	21 horas	lun 05/02/24	mié 07/02/24	
Investigación Business Central	14 horas	jue 08/02/24	vie 09/02/24	3
Módulos Business Central	42 horas	lun 12/02/24	lun 19/02/24	4
AL (Application Language)	42 horas	mar 20/02/24	mar 27/02/24	5
Planificación y análisis	**8 horas**	**mié 28/02/24**	**jue 29/02/24**	**2**
Definición objetivos y alcance	2 horas	mié 28/02/24	mié 28/02/24	
Requisitos a cumplir	2 horas	mié 28/02/24	mié 28/02/24	8
Estudio herramientas a usar	4 horas	mié 28/02/24	jue 29/02/24	9
Inicialización	**7 horas**	**jue 29/02/24**	**jue 29/02/24**	**7**
Creación y configuración proyecto en Visual Studio Code	3 horas	jue 29/02/24	jue 29/02/24	
Creación y configuración app en Power Apps	4 horas	jue 29/02/24	jue 29/02/24	12
Desarrollo	**119 horas**	**vie 01/03/24**	**jue 28/03/24**	**11**
Creación aplicación en PA	**56 horas**	**vie 01/03/24**	**mié 13/03/24**	
Creación formulario	14 horas	vie 01/03/24	mar 05/03/24	
Control de cámara	7 horas	mié 06/03/24	mié 06/03/24	16
Guardar foto	7 horas	jue 07/03/24	jue 07/03/24	17
Botón envío de datos	7 horas	vie 08/03/24	vie 08/03/24	18
Implementación API BC	21 horas	lun 11/03/24	mié 13/03/24	19
Gastos auxiliares	**63 horas**	**jue 14/03/24**	**jue 28/03/24**	**15**
Creación estructura de datos	21 horas	jue 14/03/24	mié 20/03/24	
Inserción datos en BC	21 horas	jue 21/03/24	lun 25/03/24	22
Modificación datos en BC	21 horas	mar 26/03/24	jue 28/03/24	23
Diseño	**14 horas**	**mar 02/04/24**	**mié 03/04/24**	**14**
Perfilar IU	7 horas	mar 02/04/24	mar 02/04/24	
Pop-ups de confirmación	7 horas	mié 03/04/24	mié 03/04/24	26
Pruebas	**34 horas**	**jue 04/04/24**	**mié 10/04/24**	**25**
Pruebas de aceptación	14 horas	jue 04/04/24	vie 05/04/24	
CI/CD en Github	20 horas	lun 08/04/24	mié 10/04/24	29

Figura 2.1: Diagrama Gantt de la planificación inicial.

La estimación temporal realizada se ha hecho teniendo en cuenta que la duración del proyecto es de 300 horas y que se dedican 5 días a la semana, en los que se trabaja 7 horas al día. El proyecto se ha realizado siguiendo el calendario de la empresa, así que en la planificación se tienen en consideración los días festivos. Comenzando así el proyecto el día 5 de febrero de 2024 y finalizándolo el 10 de abril de 2024.

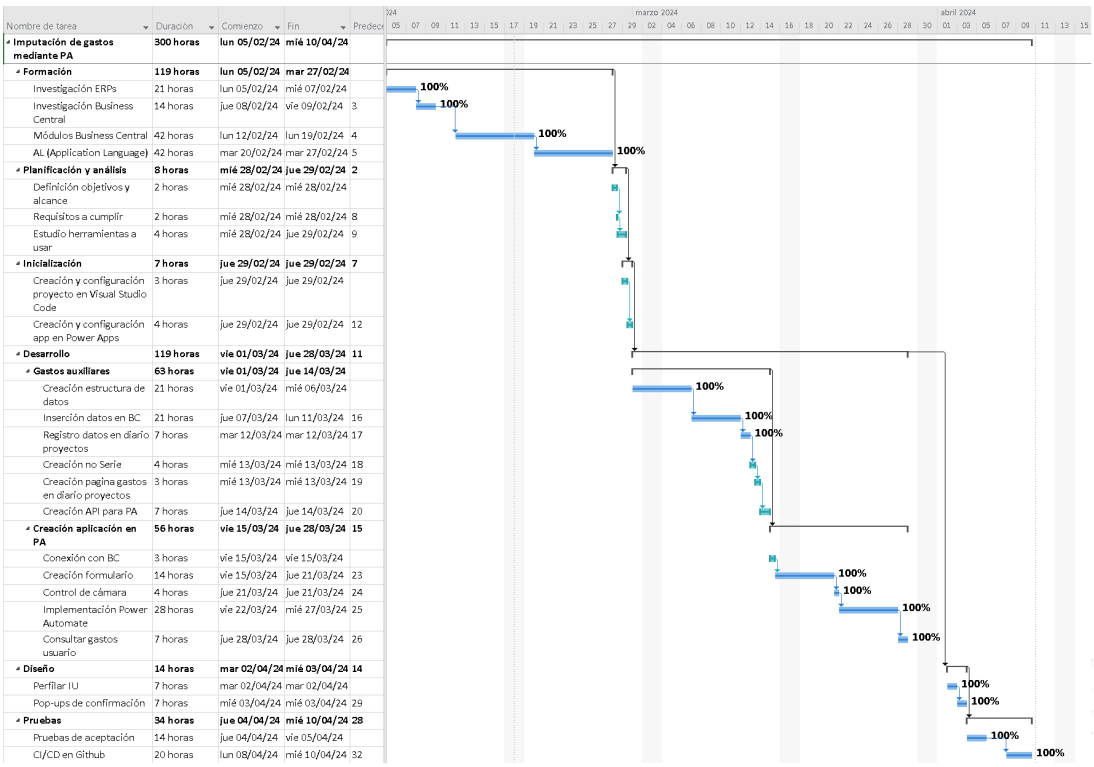
2.3. Seguimiento del proyecto

Respecto a la planificación inicial observada en la figura 2.1, como se puede observar en la figura 2.2, los cambios han sido menores; se ha mantenido el ritmo inicial, acabando así el proyecto en las 300 horas. En lo que respecta a la estructura y al orden de realización de las tareas, se observa un cambio grande: las tareas de Creación aplicación PA y Gastos auxiliares han sido invertidas. Esto se debe a que, para la realización de la aplicación en Power Apps, se requería una licencia en la nube de Business Central. Hasta el momento, se trabajaba en una versión local, y era necesario al principio configurar la aplicación conectándome con el entorno de BC. Hasta que no obtuve estos permisos, decidí comenzar por la creación de todo lo relacionado con los gastos auxiliares en BC, desde su estructura de datos, las distintas páginas y más, así estaría todo preparado para cuando pudiera acceder a PA y conectarme correctamente con BC. Además de esta modificación, también se han añadido varias subtareas necesarias, al igual que otras se han especificado algo más y otras han desaparecido. Estas modificaciones se explican más adelante.

En la tarea de Gastos auxiliares, la tarea inicial de Modificación de datos ha sido omitida, ya que una vez desarrollada la tarea de Inserción de datos en BC, automáticamente estos se pueden modificar. Esto dejaba un hueco de 21 horas que se ha llenado con la realización de las subtareas de Registro diario proyectos, Creación no Serie, Creación página gastos diario proyectos y Creación API para PA, tareas que fueron surgiendo a medida que se desarrollaba el proyecto y que eran necesarias para el correcto funcionamiento del producto.

Por último, en la tarea de Creación aplicación PA se añadió la subtarea de Conexión con BC, esta tarea se implementó tras averiguar que la conexión con BC para la lectura de datos no era necesario mediante una API. Se eliminó la subtarea Guardar foto, ya que ésta se realiza en la subtarea Control de cámara, la cual redujo su duración a 4 horas. También se eliminó la subtarea de Botón envío de datos, ya que el botón como objeto ya se incluía en la subtarea de Creación formulario, pero la función de envío la realizaba la subtarea Implementación Power Automate, la cual, por el desconocimiento de la herramienta de Power Automate, añadió unas 7 horas más de trabajo. En la planificación inicial se conocía a esta subtarea como Implementación API BC. Por último, se añadió la subtarea de Consultar gastos usuario, tarea que no se contemplo en la planificación inicial y tras una reunión con el supervisor decidió implementarse. En esta funcionalidad se filtra la información por el usuario actual, concluyendo en unas 7 horas de trabajo.

En resumen, tan solo las tareas de Desarrollo sufrieron cambios, pero en ningún momento llegó a producirse un retraso que aumentara las horas del proyecto y todo lo demás ocurrió tal cual estaba planeado.



Nombre de tarea	Duración	Comienzo	Fin	Predece
Imputación de gastos mediante PA	**300 horas**	**lun 05/02/24**	**mié 10/04/24**	
Formación	**119 horas**	**lun 05/02/24**	**mar 27/02/24**	
Investigación ERPs	21 horas	lun 05/02/24	mié 07/02/24	
Investigación Business Central	14 horas	jue 08/02/24	vie 09/02/24	3
Módulos Business Central	42 horas	lun 12/02/24	lun 19/02/24	4
AL (Application Language)	42 horas	mar 20/02/24	mar 27/02/24	5
Planificación y análisis	**8 horas**	**mié 28/02/24**	**jue 29/02/24**	**2**
Definición objetivos y alcance	2 horas	mié 28/02/24	mié 28/02/24	
Requisitos a cumplir	2 horas	mié 28/02/24	mié 28/02/24	8
Estudio herramientas a usar	4 horas	mié 28/02/24	jue 29/02/24	9
Inicialización	**7 horas**	**jue 29/02/24**	**jue 29/02/24**	**7**
Creación y configuración proyecto en Visual Studio Code	3 horas	jue 29/02/24	jue 29/02/24	
Creación y configuración app en Power Apps	4 horas	jue 29/02/24	jue 29/02/24	12
Desarrollo	**119 horas**	**vie 01/03/24**	**jue 28/03/24**	**11**
Gastos auxiliares	**63 horas**	**vie 01/03/24**	**jue 14/03/24**	
Creación estructura de datos	21 horas	vie 01/03/24	mié 06/03/24	
Inserción datos en BC	21 horas	jue 07/03/24	lun 11/03/24	16
Registro datos en diario proyectos	7 horas	mar 12/03/24	mar 12/03/24	17
Creación no Serie	4 horas	mié 13/03/24	mié 13/03/24	18
Creación pagina gastos en diario proyectos	3 horas	mié 13/03/24	mié 13/03/24	19
Creación API para PA	7 horas	jue 14/03/24	jue 14/03/24	20
Creación aplicación en PA	**56 horas**	**vie 15/03/24**	**jue 28/03/24**	**15**
Conexión con BC	3 horas	vie 15/03/24	vie 15/03/24	
Creación formulario	14 horas	vie 15/03/24	jue 21/03/24	23
Control de cámara	4 horas	jue 21/03/24	jue 21/03/24	24
Implementación Power Automate	28 horas	vie 22/03/24	mié 27/03/24	25
Consultar gastos usuario	7 horas	jue 28/03/24	jue 28/03/24	26
Diseño	**14 horas**	**mar 02/04/24**	**mié 03/04/24**	**14**
Perfilar IU	7 horas	mar 02/04/24	mar 02/04/24	
Pop-ups de confirmación	7 horas	mié 03/04/24	mié 03/04/24	29
Pruebas	**34 horas**	**jue 04/04/24**	**mié 10/04/24**	**28**
Pruebas de aceptación	14 horas	jue 04/04/24	vie 05/04/24	
CI/CD en Github	20 horas	lun 08/04/24	mié 10/04/24	32

Figura 2.2: Diagrama Gantt de seguimiento.

2.4. Costes

En esta parte se realiza una estimación de los costes que genera la realización del proyecto. Se ha decidió dividir los costes en costes de software, hardware y humanos y por último la suma de los tres.

2.4.1. Recursos hardware

Para la realización del proyecto se me ha puesto a disposición un portátil Medion Akoya E15423 con un procesador Intel Core i5-1155G7 (4 núcleos, 8 hilos, hasta 4.5GHz), 16GB de RAM, un almacenamiento de 512GB SSD y una pantalla Full HD de 15.6". Además para facilitar tanto el uso como el transporte del dispositivo se ha proporcionado un ratón inalámbrico y una mochila. Los costes de los productos se han calculado en base al uso dado, indicando su vida útil y cual es el coste en la duración del proyecto. Los costes pueden observarse en la tabla 2.1.

Costes hardware			
Producto	Precio(€)	Vida útil(años)	Coste total(€)
Portátil	599,00	5	29,95
Ratón	8,00	3	0,70
Mochila	20,00	7	0,71
Total			**31,36**

Cuadro 2.1: Coste del hardware

2.4.2. Recursos software

Para poder utilizar y desarrollar en Business Central [6] es necesario contar con una licencia activa. Estas licencias suelen ser por usuario al mes. En lo que se refiere a Power Apps [2], también es necesario una licencia, ya que se utilizan conectores premium. Para Power Automate no es necesario una licencia, ya que aunque creemos un flujo con conectores premium, si tenemos la licencia de Power Apps no habrá problema. Esto es gracias a que ambas herramientas pertenecen a Microsoft Power Platform. Los precios de las licencias utilizadas se reflejan en la tabla 2.2. Hay que indicar que el precio de la licencia es siempre por usuario y que se observa una columna con el precio por 3 meses, ya que esta es la duración del proyecto y el tiempo por el cual se necesitará la licencia activa.

Costes software		
Producto	Precio/Mes(€)	Coste total(€)
Microsoft Dynamics 365 Business Central	93,60	280,80
Microsoft Power Apps	4,70	14,10
Microsoft Power Automate	0,00	0,00
Total	**98,30**	**294,90**

Cuadro 2.2: Coste del software

2.4.3. Recursos humanos

Para el desarrollo del proyecto se necesita un empleado encargado de la formación y un analista, quien ha estado junto al desarrollador en los primeros compases del proyecto para estudiar las necesidades del cliente y diseñar una solución. Por último, está el desarrollador de software que se encarga de desarrollar la solución. En la tabla 2.3 se encuentra un desglose de lo que cobran los empleados por hora y las horas realizadas en el proyecto.

Costes recursos humanos			
Rol	Sueldo/Hora(€)	Horas	Coste total(€)
Empleado(Formación)	14,00	42	588,00
Analista	15,83	90	1424,70
Desarrollador junior	6,00	300	1800,00
Total			**3812,70**

Cuadro 2.3: Coste de recursos humanos

A los costes mostrados en la tabla 2.3 hay que sumarles tanto costes de contratación, los

cuales oscilan entre el 20 % y 30 %, como los costes indirectos, que son el 20 %. Por lo tanto, el gasto total en recursos humanos es de:

$$3812{,}70 + 25\,\%(\text{Gastos contratación}) + 20\,\%(\text{Gastos indirectos}) = 5528{,}42€$$

2.4.4. Costes totales

Teniendo en cuenta el coste total de los anteriores apartados, podemos decir que el proyecto tendrá un coste total de 6450,32€, como se indica en la tabla 2.4.

Costes totales	
Tipo coste	Total(€)
Hardware	627,00
Software	294,90
Recursos humanos	3812,70
Total	**6450,32**

Cuadro 2.4: Coste proyecto

2.5. Riesgos

En el siguiente apartado se han identificado los posibles riesgos que se podrán encontrar durante la elaboración del proyecto

2.5.1. Identificación de los riesgos

Los riesgos que podemos encontrar durante el transcurso del proyecto son los siguientes:

- R01: Falta de experiencia en el ERP Business Central
- R02: Falta de experiencia en el lenguaje AL
- R03: Falta de experiencia en Microsoft Power Apps
- R04: Desconocimiento sobre como transferir datos entre Power Apps y Business Central
- R05: Falta de tiempo

2.5.2. Análisis del riesgo

Cada uno de los riesgos anteriormente definidos serán definidos, analizados y descritos como se puede observar en el cuadro 2.5. Para cada uno de los riesgos se indicará su nivel de magnitud, su descripción, el impacto que tiene, indicadores de alerta y el tipo de riesgo.

Id	Análisis del riesgo
R01	**Magnitud:** Bastante alta **Descripción:** Poco conocimiento de como funciona el módulo donde se va a interactuar, para esto es necesario conocimientos sobre el funcionamiento del ERP. **Impacto:** Podría originar retrasos en la realización del proyecto, o que el producto tenga errores. **Indicadores:** No procede. **Tipo:** Riesgo del producto/proyecto
R02	**Magnitud:** Alta **Descripción:** Se desconoce el lenguaje y la información sobre este es escasa. **Impacto:** Podría originar retrasos en la realización del proyecto, o que el producto tenga errores. **Indicadores:** No procede. **Tipo:** Riesgo del producto/proyecto
R03	**Magnitud:** Media **Descripción:** Se desconoce la herramienta y su funcionamiento. **Impacto:** Podría originar retrasos en la realización del proyecto, o que la aplicación sea lanzada con errores. **Indicadores:** No procede. **Tipo:** Riesgo del producto/proyecto
R04	**Magnitud:** Alta **Descripción:** Se desconoce como conectar PA con BC, y la variedad de soluciones es grande, sin saber cual es la adecuada. **Impacto:** Podría originar retrasos en la realización del proyecto. **Indicadores:** No procede. **Tipo:** Riesgo del proyecto
R05	**Magnitud:** Media **Descripción:** El proyecto tiene unos plazos y tiempo a cumplir. **Impacto:** Podrían surgir perdidas de dinero. **Indicadores:** No procede. **Tipo:** Riesgo del proyecto

Cuadro 2.5: Análisis del riesgo

2.5.3. Acciones de Prevención y Corrección

Para finalizar, en el cuadro 2.6 se especifican los distintos planes tanto de prevención como de corrección planeados por si alguno de los riesgos ocurriera.

Id	Plan de prevención/acción	Plan de corrección/contingencia
R01	Realizar un formación extensa sobre todos los módulos del ERP, poniendo hincapié en aquellos que vamos a utilizar.	Pausar y pedir ayuda a alguno de los consultores funcionales de la empresa, para que nos expliquen con detalle cómo funciona el módulo y cómo debemos interactuar con él.
R02	Realizar una formación intensa, sobre cómo usar este lenguaje y su alcance funcional	Pausar y revisar la documentación de Microsoft o preguntar al supervisor de la empresa.
R03	Realizar una formación sobre el funcionamiento básico de Power Apps y su alcance funcional.	Pausar y revisar la documentación, mucha de ella de creadores de contenido de esta herramienta. Si no, consultar con aquellos empleados que hayan utilizado Power Apps.
R04	Discutir cual es la solución mas factible para el traspaso de datos, y documentarse sobre cómo hacerla.	Si la solución elegida no llega a ser la más factible, buscar otra y consultarlo con el supervisor en la empresa.
R05	Realizar desde el primer día una planificación inicial del tiempo	Recortar en aquellas funcionalidades que no sean necesarias para el funcionamiento básico.

Cuadro 2.6: Plan de prevención y corrección

Capítulo 3

Análisis del sistema/producto

3.1. Definición de requisitos

Partiendo del requerimiento funcional del cliente, se han identificado los distintos requisitos que necesitara el sistema para cumplir los objetivos propuestos. Para la especificación de los requisitos se ha desarrollado un diagrama de casos de usos, donde los casos de usos representan las funcionalidades que se le ofrecerá al cliente y las requeridas para el correcto funcionamiento del sistema. También se pueden identificar varios actores, que corresponden a servicios o usuarios que realizaran los casos de usos relacionados. Todo esto se puede observar en el diagrama 3.1.

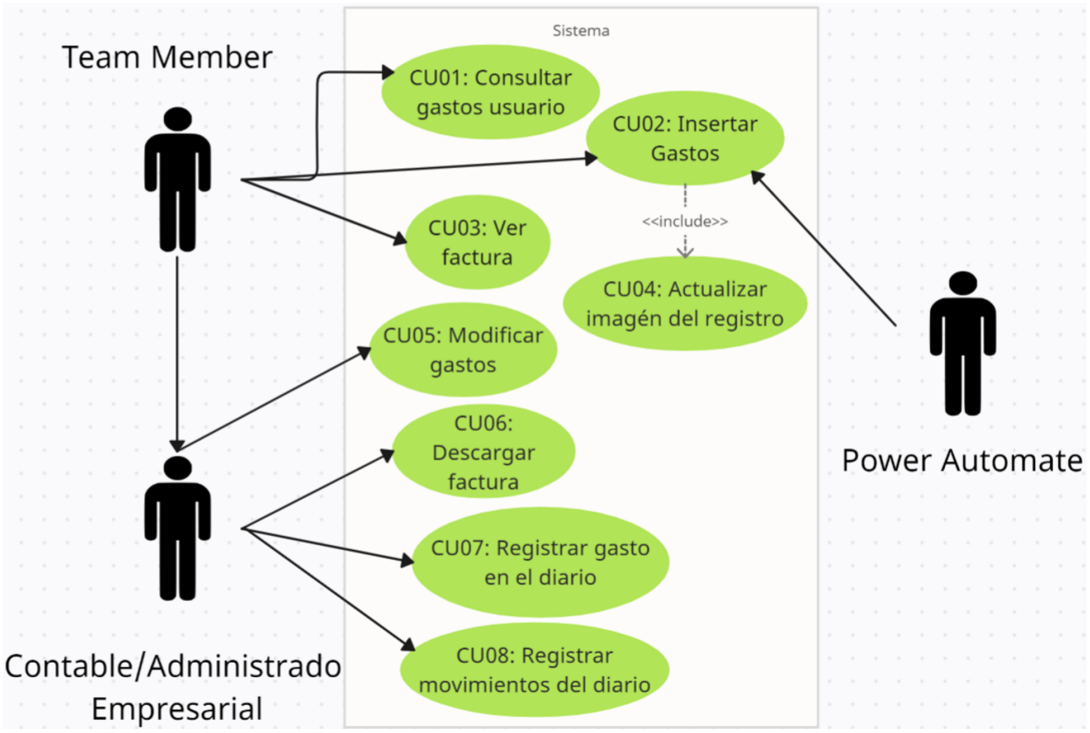


Figura 3.1: Diagrama casos de uso

Como se observa, el diagrama 3.1 cuenta con tres actores los cuales se detallan en más profundidad en el cuadro 3.1:

Id	Actor	Descripción
A01	Contable/Administrador empresarial	El contable o administrador principal es el encargado de registrar el dato definitivamente al proyecto, pudiendo también hacer uso de la aplicación móvil, se podría decir que tiene acceso a todas las funcionalidades del sistema.
A02	Team member	El team member es un usuario con funcionalidades limitadas que tan solo puede utilizar la aplicación móvil.
A03	Power Automate	Power Automate es el servicio que se encargará de insertar los datos desde la app móvil a Business Central, más tarde coge el identificador del registro que acaba de insertar y añade la imagen de la factura.

Cuadro 3.1: Definición de los actores

A continuación, se detallan los casos de uso del sistema incluyendo así su descripción y sus secuencias de uso tanto las correctas como si existen incorrectas.

Identificador	**CU01**
Nombre:	Consultar gastos usuario
Versión:	1.0
Fuente:	Pau Castejón Serrano
Actor principal:	Team Member
Descripción	Este caso de uso representa la funcionalidad de consultar los gastos que has insertado.
Secuencia normal	1. El usuario entra en la aplicación 2. El usuario pulsa el botón de consultar gastos 3. El sistema muestra el listado de gastos
Excepciones	1. El usuario entra en la aplicación 2. El usuario pulsa el botón de consultar gastos 3. El sistema no muestra ningún gasto ya que no existen

Cuadro 3.2: Definición del CU01

Identificador	**CU02**
Nombre:	Insertar gastos
Versión:	1.0
Fuente:	Pau Castejón Serrano
Actor principal:	Team Member
Descripción	Este caso de uso representa la funcionalidad de insertar gastos en el listado de gastos de Business Central.
Secuencia normal	1. El usuario entra en la aplicación 2. El usuario pulsa el botón de añadir gastos 3. El sistema muestra el formulario 4. El usuario llena el formulario 5. El usuario adjunta una foto del ticket 6. El usuario le da al botón de insertar 7. El sistema inserta los gastos
Secuencia alternativa	1. El usuario entra en la aplicación 2. El usuario pulsa el botón de consultar gastos 3. El sistema muestra el listado de gastos 4. El usuario presiona el botón de añadir 5. El sistema muestra el formulario 6. El usuario llena el formulario 7. El usuario adjunta una foto del ticket 8. El usuario le da al botón de insertar 9. El sistema inserta los gastos
Excepciones	1. El usuario entra en la aplicación 2. El usuario pulsa el botón de añadir gastos 3. El sistema muestra el formulario 4. El usuario llena el formulario 5. El usuario le da al botón de insertar 6. El sistema muestra una excepción ya que falta por adjuntar el ticket

Cuadro 3.3: Definición del CU02

Identificador	**CU03**
Nombre:	Ver factura
Versión:	1.0
Fuente:	Pau Castejón Serrano
Actor principal:	Team Member
Descripción	Este caso de uso representa la funcionalidad de poder consultar la imagen adjunta(factura).
Secuencia normal	1. El usuario entra en la aplicación 2. El usuario pulsa el botón de consultar gastos 3. El sistema muestra el listado de gastos 4. El usuario entra en un gasto en concreto 5. El usuario persiona el botón de ver factura 6. El sistema muestra la factura por pantalla.

Cuadro 3.4: Definición del CU03

Identificador	**CU04**
Nombre:	Actualizar imagen del registro
Versión:	1.0
Fuente:	Pau Castejón Serrano
Actor principal:	Power Automate
Descripción	Este caso de uso representa la funcionalidad como Power Automate inserta una imagen en Business Central.
Secuencia normal	1. El sistema tras la inserción busca el gasto por su ID 2. El sistema inserta al campo Image la imagen adjuntada

Cuadro 3.5: Definición del CU04

Identificador	**CU05**
Nombre:	Modificar gastos
Versión:	1.0
Fuente:	Pau Castejón Serrano
Actor principal:	Contable/Administrador empresarial
Descripción	Este caso de uso representa la funcionalidad de modificar los gastos que ya están en BC.
Secuencia normal	1. El usuario accede a la lista de gastos desde Business Central 2. El usuario pulsa en el gasto a modificar 3. El usuario modifica los datos que sean necesarios 4. El sistema actualiza automáticamente los datos
Excepciones	1. El usuario accede a la lista de gastos desde Business Central 2. El usuario pulsa en el gasto a modificar 3. El usuario no puede modificar ningún dato ya que el gasto esta registrado 1. El usuario accede a la lista de gastos desde Business Central 2. El usuario pulsa en el gasto a modificar 3. El usuario modifica el nombre del documento 4. El sistema lanza una excepción ya que el nombre ya existe y no modifica ningún dato.

Cuadro 3.6: Definición del CU05

Identificador	**CU06**
Nombre:	Descargar factura
Versión:	1.0
Fuente:	Pau Castejón Serrano
Actor principal:	Contable/Administrador empresarial
Descripción	Este caso de uso representa la funcionalidad de poder descargar la factura desde BC.
Secuencia normal	1. El usuario accede a la lista de gastos desde Business Central 2. El usuario pulsa en el gasto que desee 3. El usuario pulsa el botón de Descargar Factura 4. El sistema descarga en un archivo .jpg la factura
Excepciones	1. El usuario accede a la lista de gastos desde Business Central 2. El usuario pulsa en el gasto que desee 3. El usuario pulsa el boton de Descargar Factura 4. El sistema suelta un excepción ya que no hay factura que descargar

Cuadro 3.7: Definición del CU06

Identificador	**CU07**
Nombre:	Registrar gasto en el diario general de proyectos
Versión:	1.0
Fuente:	Pau Castejón Serrano
Actor principal:	Contable/Administrador empresarial
Descripción	Este caso de uso representa la funcionalidad de poder registrar en el diario general de proyectos el o los gastos.
Secuencia normal	1. El usuario accede a la lista de gastos desde Business Central 2. El usuario pulsa en el gasto a registrar 3. El usuario pulsa el botón de Registrar en el diario 4. El sistema registra el gasto en el diario
Secuencia alternativa	1. El usuario accede a la lista de gastos desde Business Central 2. El usuario selecciona varios gastos a la vez para registrarlos 3. El usuario pulsa el botón de Registrar en el diario 4. El sistema registra el gasto en el diario
Excepciones	1. El usuario accede a la lista de gastos desde Business Central 2. El usuario entra en el gasto a registrar 3. El usuario pulsa el botón de Registrar en el diario 4. El sistema suelta un excepción ya que el gasto ya esta registrado

Cuadro 3.8: Definición del CU07

Identificador	**CU08**
Nombre:	Registrar movimientos del diario general de proyectos
Versión:	1.0
Fuente:	Pau Castejón Serrano
Actor principal:	Contable/Administrador empresarial
Descripción	Este caso de uso representa la funcionalidad de como los gastos son registrados en los proyectos correspondientes junto a su factura.
Secuencia normal	1. El usuario accede a la lista de gastos desde Business Central 2. El usuario pulsa en el botón de Diario de proyectos 3. El usuario pulsa el botón de Registrar 4. El sistema registra el gasto en el proyecto junto a su factura
Excepciones	1. El usuario accede a la lista de gastos desde Business Central 2. El usuario pulsa en el botón de Diario de proyectos 3. El usuario pulsa el botón de Registrar 4. El sistema genera una excepción indicado la linea y el error

Cuadro 3.9: Definición del CU08

Tras la definición de los casos de usos, en las siguientes tablas se encuentran los requisitos demandados por el sistema y que el usuario deberá introducir para el correcto funcionamiento de este.

Identificador	**RD01**
Nombre: Descripción	Gasto Requisito que representa los datos sobre un nuevo gasto.
Datos específicos	Nombre del documento, imagen, nº proyecto, nº tarea, asunto, descripción, coste, código divisa, nº cuenta, tipo de contrapartida, nº cuenta contrapartida, fecha factura.
Detalles	**Nombre del documento:** se asigna automáticamente al añadir al gasto siguiendo el último número de un número de serie previamente creado. El nombre del documento puede ser modificado. **Imagen:** puede realizarse una foto o subirla desde los archivos del dispositivo. **Nº proyecto:** se selecciona de un desplegable de los proyectos existentes. **Nº tarea:** se selecciona de un desplegable mostrando las tareas del proyecto anteriormente escogido. **Código divisa:** se selecciona de un desplegable que contiene todas las divisas registradas en Business Central. No rellenar este campo significa elegir la divisa de tu región actual. **Nº cuenta**: se selecciona de un desplegable que contiene las cuentas del plan de cuentas que sean de gastos. **Tipo de contrapartida:** se escoge de un desplegable el tipo de contrapartida. **Nº cuenta contrapartida:** se selecciona ya sea una cuenta, un banco o una persona que corresponda al tipo de contrapartida que hayamos escogido. **Fecha factura:** se selecciona de un calendario la fecha a escoger.

Cuadro 3.10: Definición del RD01

En algunos casos es importante documentar requisitos relacionados que no corresponden directamente con la funcionalidad del sistema. En el cuadro 3.11 se define uno de estos:

Identificador	**RA01**
Nombre: Descripción	Velocidad de carga. Tanto la aplicación móvil como la parte de Business Central deben ser lo suficientemente rápidas para que puedan ser utilizadas cómodamente. La parte de la app móvil se le exigirá un tiempo de respuesta menor a 5 segundos cuando se quiera insertar un dato u observar la lista de gastos. La parte de Business Central debería tardar como mucho 2 segundos por gasto cuando se registren en el diario y menos de un segundo en descargar la factura.

Cuadro 3.11: Definición del RA01

3.2. Diagrama de clases

Una vez realizada la definición de requisitos hay que comprender que es lo que hace el sistema para cumplir con estos. Para ello se ha realizado un diagrama de clases que muestra las clases que forman parte de la solución y cómo interactúan entre ellas. Como se puede observar en la figura 3.2, no se han detallado los atributos ya que estos se muestran en el siguiente apartado referente al diseño de la base de datos.

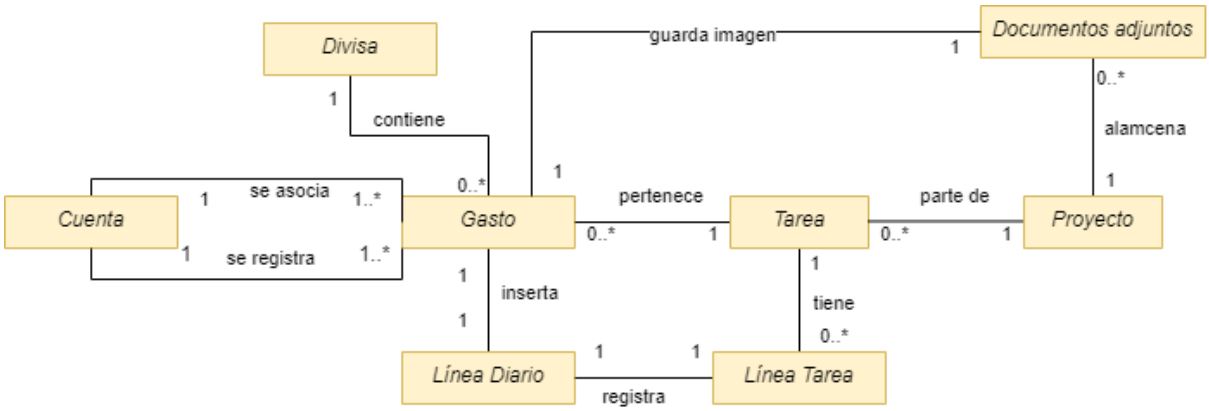


Figura 3.2: Diagrama de clases del sistema

Como se observa en el diagrama, la clase Gasto es la que llega a interactuar con todo el sistema, en ciertas clases como Divisa o Cuenta, por el hecho de almacenar datos de estas. En otras clases como Línea Diario o Documentos Adjuntos, por el hecho de insertar datos en estas. Se podría llegar a la conclusión de que la clase Gasto es el núcleo de la solución.

Capítulo 4

Diseño del sistema/producto

4.1. Diseño de la base de datos

Al haber desarrollado este proyecto en torno al ERP de Microsoft Business Central, la base de datos ya venía configurada y se trata de SQL Server [7]. Tan solo desde nuestro entorno de programación, una vez conectados a Business Central, podríamos acceder a todas las tablas creadas y crear nuevas. En la figura 4.1 se puede observar la arquitectura y cómo está conectada la base de datos.

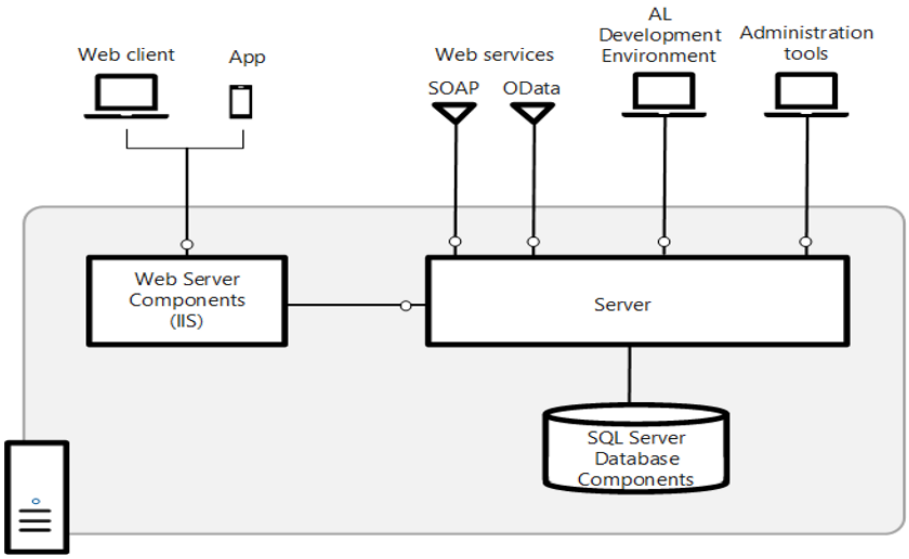


Figura 4.1: Representación de la interacción con la base de datos [4]

En lo que respecta a Power Apps, los datos se almacenan en tablas que se conectan a Business Central y recogen todos los datos de la tabla que elijamos. Los datos se muestran por pantalla mediante unas estructuras de datos llamadas galerías. Estas estructuras se asocian con la tabla de datos que escojamos pudiendo seleccionar qué campos enseñar o no.

En Business Central se ha creado una tabla nueva llamada *Cost*, la cual se relaciona con otras tablas para obtener datos necesarios. La tabla se puede observar en la figura 4.2.

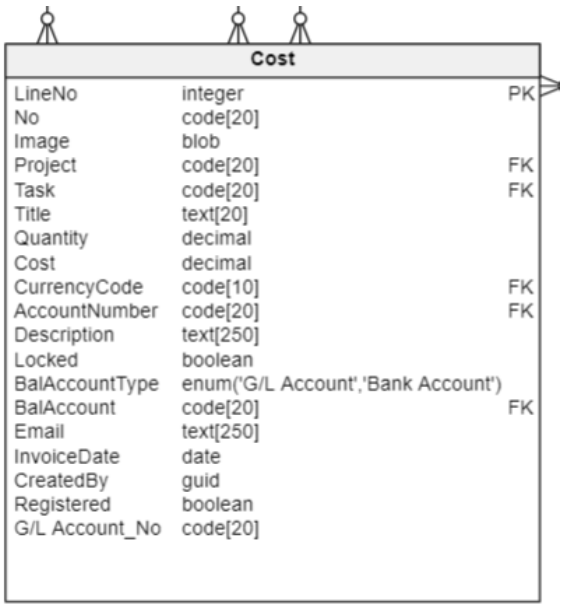


Figura 4.2: Definición de la tabla Cost

Para registrar los gastos, se ha decidido insertarlos previamente en la tabla *Gen. Journal Line*. Esta tabla corresponde a las líneas insertadas en el Diario general de proyectos, donde una vez almacenados los gastos se procede a registrarlos. Los gastos se registran en la tabla *Job Planning Line*, donde se almacenan todas las líneas relacionadas con los proyectos y tareas. El ticket correspondiente del gasto se inserta con un nombre que ayuda a identificar a qué tarea corresponden en la tabla *Document Attachment*, donde se almacenan todos los documentos adjuntos del ERP. En la siguiente figura 4.3 se puede observar el diseño físico de la base de datos.

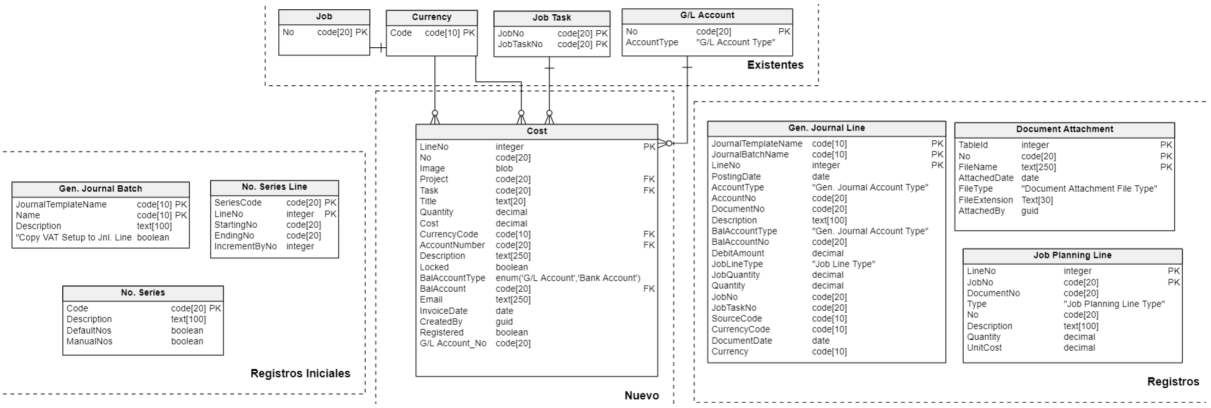


Figura 4.3: Diseño físico de la base de datos

En el diagrama se pueden observar distintas áreas. La existente corresponde a tablas que forman parte ya de la base de datos. Éstas han sido insertadas por Microsoft. Dado que estas tablas tienen más de 100 columnas, se ha decidido mostrar tan solo las claves primarias. Como se observa, estas tablas ayudan a relacionar el gasto con el proyecto y con varios datos fiscales.

Las tablas del área de registros iniciales corresponden a inserciones que se realizan al instalar la aplicación, para insertar de manera automática el número de serie que utilizará nuestra tabla de gastos y una sección en el diario general de proyectos llamada "gastos". En esta sección siempre se insertarán nuestros gastos para diferenciarlos del resto de información que se añada al diario general de proyectos.

En el área cuyo nombre es nuevo, se indican las tablas creadas; en este caso, la tabla *Cost* que corresponde a los gastos.

Por último, en la zona de registros se observan las tablas en las que se almacenan los datos de los gastos, como se indicó anteriormente.

4.2. Diseño de la arquitectura del sistema/producto

La arquitectura del sistema define sus componentes principales, las relaciones entre estos y cómo interactúan. En este caso, y tal como se observa en el esquema general de la figura 4.4, nuestro sistema corresponde a una arquitectura de 3 capas.

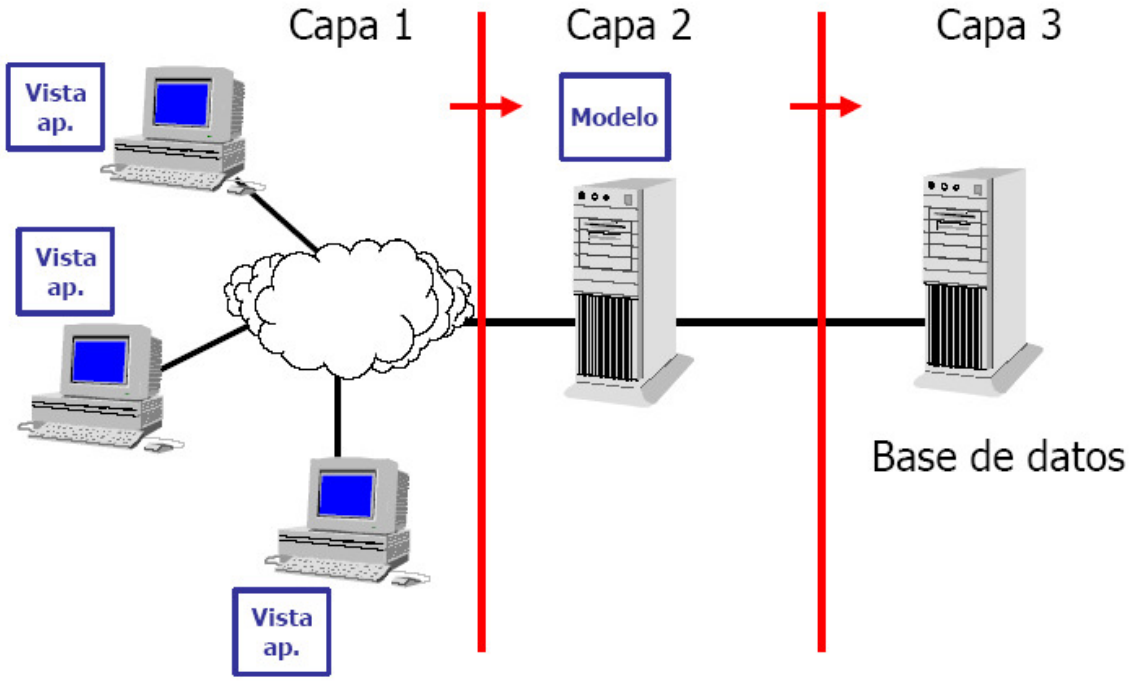


Figura 4.4: Esquema de un sistema con una arquitectura de 3 capas [1].

Estas 3 capas se organizan en nuestro sistema de la siguiente manera:

- Capa de presentación: en esta capa se encuentra la interfaz del usuario con la que podremos interactuar con el sistema. Aquí entran tanto las interfaces creadas dentro del entorno de Business Central como las que forman parte de la aplicación creada en Power Apps.
- Capa de negocio: aquí es donde el sistema responde a todas las peticiones del usuario, en nuestro caso, la orden de crear un gasto, de registrarlo, de descargar la factura y todos aquellos procesos que hemos definido para el producto.
- Capa de datos: los datos que utiliza el sistema, como sería en nuestro caso los datos de la tabla gastos, los cuales se almacenan en una base de datos Microsoft SQL Server.

Para el correcto funcionamiento del sistema, este ha de estar alojado en la nube. Esto quiere decir que la versión local de Business Central no valdría. Al realizar conexiones externas con Power Apps, se han aprovechado los distintos conectores que ofrecen las herramientas de Power Platform. Estos conectores tan solo funcionan para realizar conexiones con entornos en la nube, por esto la versión de Business Central debe ser la que está alojada en la nube.

4.3. Diseño de las interfaces

A continuación se van a mostrar las distintas interfaces de usuario que permiten al usuario interactuar con el sistema. Primero se mostrarán las interfaces relacionadas con la aplicación móvil y por último las relacionadas con Business Central.

En la figura 4.5 se puede observar la página inicial, donde se muestra en una esquina el logotipo de la empresa que ha desarrollado la aplicación. También se puede ver el nombre de la aplicación, un icono de una factura y un botón Comenzar que redirigirá a la pantalla de selección.

Figura 4.5: Página inicio

Una vez le demos al botón, nos redirigirá a la pantalla mostrada en la figura 4.6, donde debemos escoger qué acción realizar. Si le damos al botón Añadir gastos, nos enviará a un formulario donde rellenar los datos a insertar; en cambio, si le damos a Consultar gastos, podremos ver un listado de los gastos que hemos imputado.

Figura 4.6: Menú de la aplicación

Si le damos a la primera opción se nos abrirá un formulario con distintos campos a rellenar, figura 4.7. Algunos campos deberán seleccionarse de una lista desplegable, como el proyecto, otros como la fecha se seleccionan a través de un calendario, y por último la imagen la cual podemos insertar haciendo una foto al momento o seleccionar algún archivo. Todo esto último se observa en la figura 4.8.

Figura 4.7: Formulario para añadir gasto

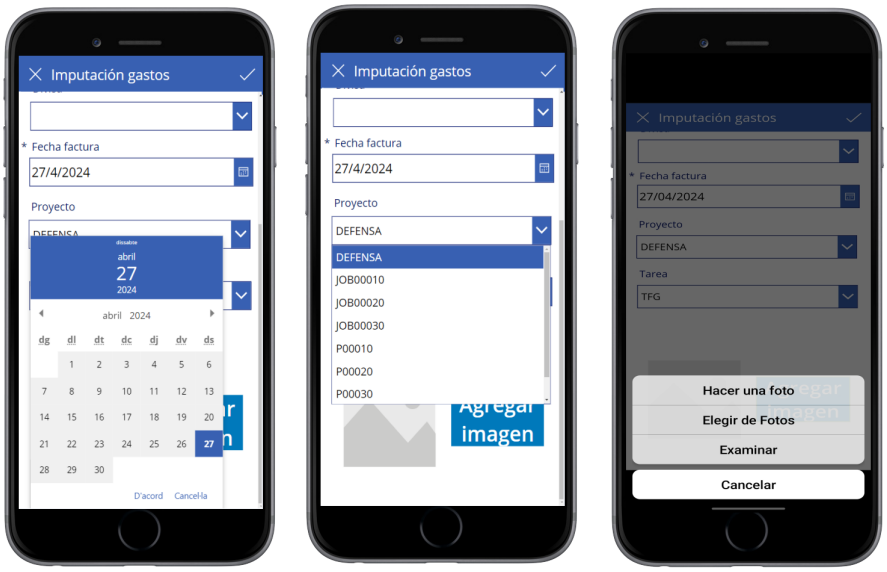


Figura 4.8: Opciones campos formularios

Al rellenar los datos y darle al botón de insertar (el símbolo del tic), nos saldrá una ventana emergente indicando si nos faltan datos por rellenar o si se han insertado correctamente. Esto se muestra en la figura 4.9.

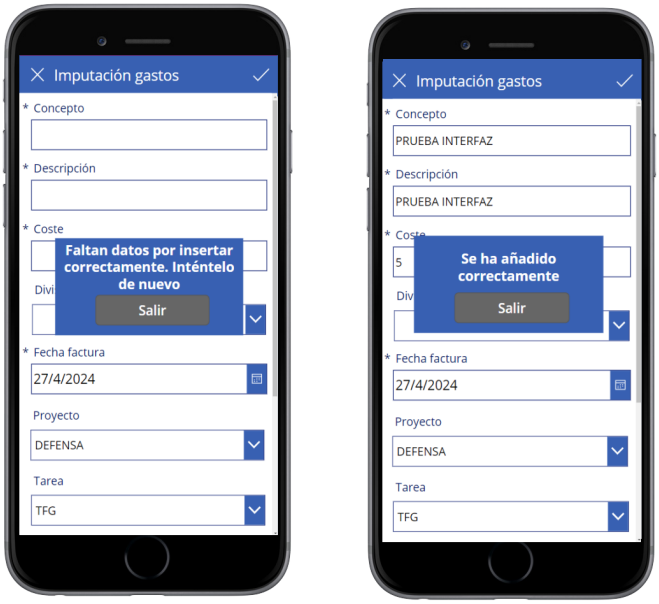


Figura 4.9: Respuesta al insertar formulario

Si al insertar correctamente le damos a Salir, nos redirigirá automáticamente al listado de gastos. A este listado también se accede dándole al botón de Consultar gastos en la figura 4.6. El listado muestra la información necesaria para identificar al gasto, como su concepto, descripción y el coste. Se observa en la figura 4.10.

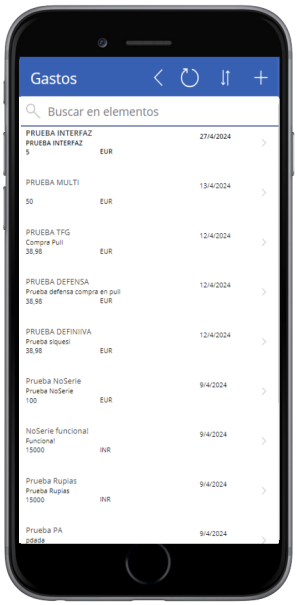


Figura 4.10: listado de gastos

Dentro de la página del listado también se pueden realizar acciones como añadir un gasto, actualizar la lista o ordenar por fecha. También podemos buscar un gasto por su nombre en la barra de búsqueda. Si seleccionamos un gasto en concreto, nos redirigirá a una pantalla donde veremos en más detalle el gasto, como se muestra en la figura 4.11.

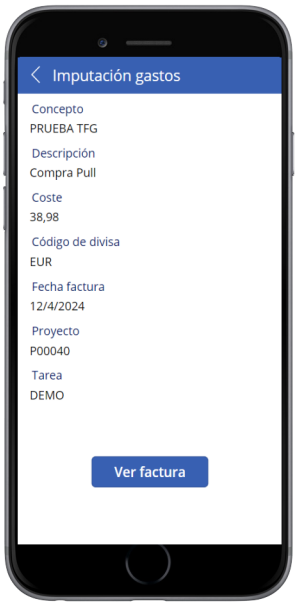
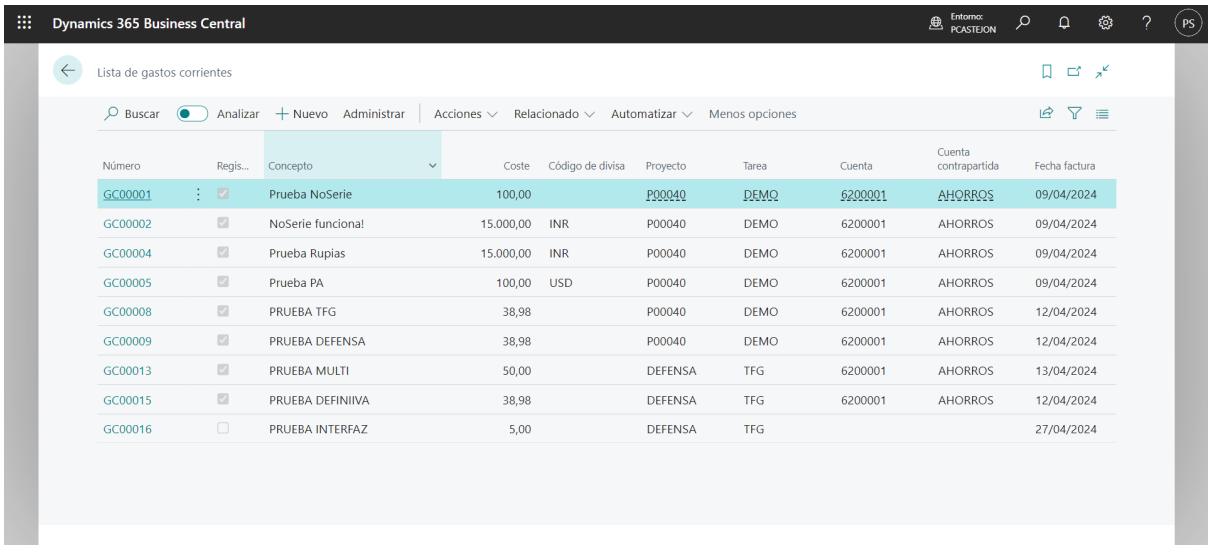


Figura 4.11: Detalles del gasto

Dentro de los detalles de un gasto, si le damos al botón "Ver factura", nos mostrará la factura por pantalla, pudiendo ampliar la imagen si es necesario. Se puede ver la factura ampliada en la figura 4.12.

Figura 4.12: Detalles factura

Una vez mostradas todas las interfaces de la aplicación móvil, pasamos a mostrar las de Business Central. Para acceder al listado en Business Central, tan solo hay que buscar en el buscador "Lista de gastos corrientes". Una vez dentro, el listado se mostrará como en la figura 4.13:



Número	Regis...	Concepto	Coste	Código de divisa	Proyecto	Tarea	Cuenta	Cuenta contrapartida	Fecha factura
GC00001	✓	Prueba NoSerie	100,00		P00040	DEMO	6200001	AHORROS	09/04/2024
GC00002	✓	NoSerie funciona!	15.000,00	INR	P00040	DEMO	6200001	AHORROS	09/04/2024
GC00004	✓	Prueba Rupias	15.000,00	INR	P00040	DEMO	6200001	AHORROS	09/04/2024
GC00005	✓	Prueba PA	100,00	USD	P00040	DEMO	6200001	AHORROS	09/04/2024
GC00008	✓	PRUEBA TFG	38,98		P00040	DEMO	6200001	AHORROS	12/04/2024
GC00009	✓	PRUEBA DEFENSA	38,98		P00040	DEMO	6200001	AHORROS	12/04/2024
GC00013	✓	PRUEBA MULTI	50,00		DEFENSA	TFG	6200001	AHORROS	13/04/2024
GC00015	✓	PRUEBA DEFINIIVA	38,98		DEFENSA	TFG	6200001	AHORROS	12/04/2024
GC00016		PRUEBA INTERFAZ	5,00		DEFENSA	TFG			27/04/2024

Figura 4.13: Listado gastos en BC

En el listado se muestra de cada gasto la información más relevante, como un tic que informa si ya ha sido registrado o no, o información fiscal que se rellena desde Business Central. Si un gasto tiene algún campo fiscal vacío, significa que hay que entrar a rellenarlo. Desde el listado se pueden ejecutar dos acciones: la de registrar gasto y la de ir al diario general de proyectos. La primera se ubica en el menú Acciones de la barra horizontal. La segunda acción está en el menú Relacionado. En la figura 4.14 se puede observar la ubicación de estos menús.

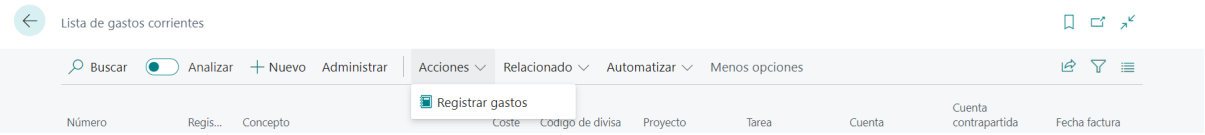


Figura 4.14: Menú acciones barra horizontal

Si hacemos clic en el número del gasto que queramos, nos redirigirá a una página donde se muestran en más detalle los datos del gasto, como se observa en la figura 4.15. Según el tic mostrado en el listado (indicando si está registrado o no), si el gasto está registrado, accederemos al gasto sin poder editarlo; de lo contrario, podremos editar cualquier dato. Al igual que en el listado, en el menú Acciones encontraremos la función para registrar el gasto, también la acción para descargar la factura, descargando un archivo .jpg en nuestro dispositivo. En el menú Relacionado encontramos la redirección al diario general de proyectos.

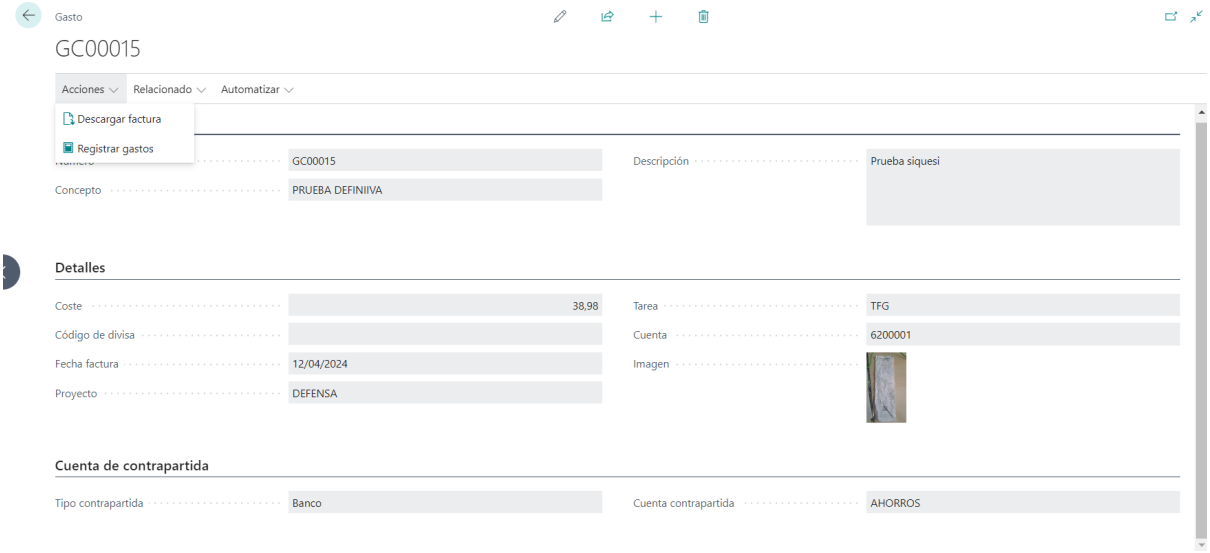


Figura 4.15: Detalles de un gasto

Por último se han diseñado dos ventanas emergentes que saltarán al ejecutar la acción de registrar según el resultado obtenido. Ambos mensajes se muestran en la figura 4.16.

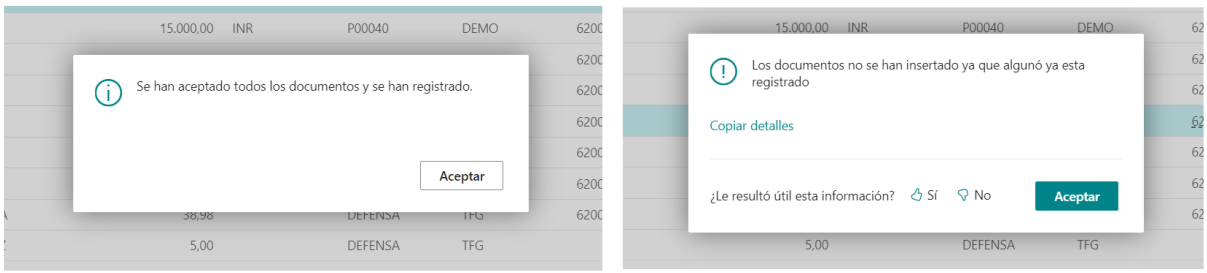


Figura 4.16: Ventanas emergentes tras registrar gasto

Capítulo 5

Implementación y pruebas

5.1. Estructura del código

La estructura del código se ha organizado por los distintos tipos de objetos que existen. La extensión en Visual Studio Code llamada *waldo's CRS AL Language Extension* ayuda a reorganizar los archivos por su tipo, manteniendo así una organización en el código. Los distintos tipos de objetos que se han utilizado son los siguientes:

- Codeunit: bloque de código donde se encapsulan funciones que se utilizarán en otros objetos.
- Enum: objeto en el que se representan un conjunto de valores constantes, para limitar las opciones de valores en una variable.
- Page: objeto que representa la interfaz del usuario. Obtiene los datos de una tabla y los muestra según el tipo de página seleccionado.
- Page Extension: objeto que representa una extensión de una página, añadiendo nuevos campos que mostrar o nuevas funcionalidades.
- Permission Set: conjunto de permisos de lectura y escritura que se han de indicar para los usuarios.
- Table: objeto que representa una tabla de la base de datos. Contiene datos y alguna funcionalidad, al igual que los disparadores que afectan a esta tabla.
- Table Extension: objeto que representa la extensión de una tabla, añadiendo nuevos campos o nuevas funcionalidades.
- Translations: ficheros de traducción, generados automáticamente por la extensión *AL Language Tools*.

Además de estos objetos, en el proyecto se encuentran otros tipos de archivos ya sean de configuración o de datos. En nuestro proyecto podemos encontrar los siguientes:

- .alpackages: en este directorio se encuentran todas las estructuras que forman parte del ERP actualmente, como páginas, tablas, codeunits...

- launch.json: archivo de configuración en el que se detalla información como el servidor donde ejecutar el código.

- app.json: fichero de configuración del proyecto, donde se detalla el nombre del editor, la empresa, versión, etc.

- Archivos .app: estos archivos se generan teniendo en cuenta la información del app.json y sirven para cargar nuestro proyecto de manera manual en entornos de Business Central.

Toda esta organización se puede observar en la figura 5.1, donde se muestran dos proyectos, uno para el código y otro para los tests. En los tests se crean codeunits de tipo test que deben ser alojadas fuera del proyecto principal.

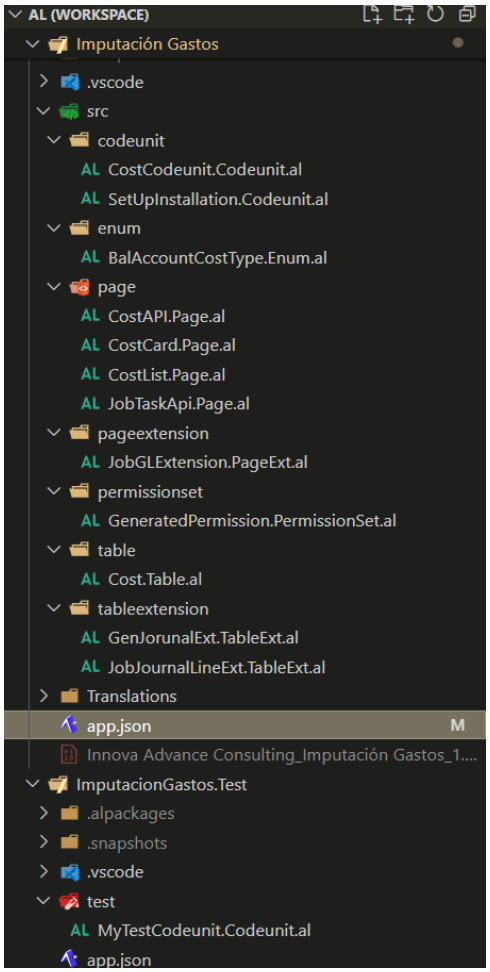


Figura 5.1: Organización del código

5.2. Descripción técnica de la implementación

En este apartado se va a explicar cómo se ha implementado el código para desarrollar el producto. Primero se hará una explicación sobre todo el código escrito en Business Central, y por último el poco código utilizado en Power Apps.

Para empezar, se ha creado un objeto de tipo tabla llamado *Cost*, en esta tabla se añadirán los gastos. La tabla se muestra en la figura 5.2.

```
table 50100 Cost
{
    DataClassification = CustomerContent;
    DataCaptionFields = No;
    You, last month | 1 author (You)
    fields
    {
        25 references
        field(1; No; Code[20])
        {       You, last month • Imputación gastos APP inserted
            Caption = 'No', Comment = 'ESP="Número"';
        }
        9 references
        field(2; Image; Blob) ···
        9 references
        field(3; Project; Code[20]) ···
        9 references
        field(4; Task; Code[20]) ···
        7 references
        field(5; Title; Text[20]) ···
        0 references
        field(6; Quantity; Decimal) ···
        7 references
        field(7; Cost; Decimal) ···
        4 references
        field(8; CurrencyCode; Code[10])
        {
            Caption = 'CurrencyCode', Comment = 'ESP="Código de divisa"';
            TableRelation = Currency;


        }
        3 references
        field(9; AccountNumber; Code[20]) ···
        5 references
        field(10; Description; Text[250]) ···
        5 references
        field(11; Locked; Boolean) ···
        4 references
        field(12; BalAccountType; Enum BalAccountCostType) ···
        3 references
        field(13; BalAccount; Code[20]) ···
        1 reference
        field(14; Email; Text[250]) ···
        field(15; InvoiceDate; Date) ···
        2 references
        field(16; CreatedBy; GUID) ···
        1 reference
        field(17; LineNo; Integer) ···
        3 references
        field(18; Registered; Boolean) ···

    }

    You, last month | 1 author (You)
    keys
    {
        0 references
        key(PK; LineNO)
        {
            Clustered = true;
        }
        0 references
        key(K1; No)
        {

        }
        0 references
        key(K2; SystemId)
        {
        }
    }
    4 references
    trigger OnInsert();
    var
        NoSeriesMgm: Codeunit NoSeriesManagement;
    begin
        Rec.No := NoSeriesMgm.GetNextNo('GCOR', WorkDate(), true);
        CreatedBy := UserSecurityId();
    end;
}
```

Figura 5.2: Tabla Cost

Cada elemento de la tabla tiene un *Caption*, que es el nombre que se mostrará. Principalmente se usa el inglés, y en un *comment*, como se observa en el primer campo, se indica el lenguaje a traducir y su traducción. Como se indicó en el anterior apartado, cuando se compile el código se generarán las traducciones de manera automática. También se ha incluido un disparador que insertará en el campo *No* el siguiente número de serie disponible. Las relaciones con otras tablas se insertan como se muestra en la figura 5.3.

```
field(8; CurrencyCode; Code[10])
{
    Caption = 'CurrencyCode', Comment = 'ESP="Código de divisa"';
    TableRelation = Currency;

}
```

Figura 5.3: Relación del campo Código de divisa con la tabla *Currency*

Para mostrar los datos hemos tenido que crear varios objetos de tipo página. En el primer caso, se ha creado una página de tipo lista para mostrar en una lista los gastos y datos que sean relevantes. En la imagen 5.4 se muestra cómo se ha creado el objeto.

```
page 50101 CostList
{
    PageType = List;
    ApplicationArea = Basic, Suite;
    UsageCategory = Lists;
    SourceTable = Cost;
    CardPageId = CostCard;
    Caption = 'Cost List', Comment = 'ESP="Lista de gastos corrientes"';
    Editable = false;
    You, last month | 1 author (You)
    layout
    {
        You, last month | 1 author (You) | 0 references
        area(Content)
        {
            0 references
            repeater(Group)
            {
                0 references
                field(No; Rec.No) ···
                0 references
                field(Registered; Rec.Registered) ···
                0 references
                field(Title; Rec.Title) ···
                0 references
                field(Cost; Rec.Cost) ···
                0 references
                field(CurrencyCode; Rec.CurrencyCode) ···
                0 references
                field(Project; Rec.Project)      You, last month • Imput
                0 references
                field(Task; Rec.Task) ···
                0 references
                field(AccountNumber; Rec.AccountNumber) ···
                0 references
                field(BalAccount; Rec.BalAccount) ···
                0 references
                field(InvoiceDate; Rec.InvoiceDate) ···
            }
        }
    }
```

Figura 5.4: Página lista de gastos

Como se observa en la figura 5.4, se indica el tipo de página en la propiedad *PageType*, la tabla que se utiliza en *SourceTable*, y en *CardPageId* una relación con el siguiente objeto de página que se ha creado para poder mostrar en más detalle los datos de un gasto. Los campos a mostrar se ubican en el área *Content*, y a cada campo se le indica con qué campo de la tabla se asocia. El segundo objeto de tipo página que creamos es de tipo Tarjeta; de esta manera, se muestran los datos en más detalle. Esta página es la que se asigna a la propiedad *CardPageId*. El código de cómo se ha implementado se muestra en la figura 5.5.

```
page 50100 CostCard
    layout
        area(Content)
            You, last month | 1 author (You) | 0 references
            group(General)
            {
                Caption = 'General', Comment = 'ESP="General"';
                0 references
                field(No; Rec.No) ···
                0 references
                field(Title; Rec.Title) ···
                0 references
                field(Description; Rec.Description) ···
            }
            You, last month | 1 author (You) | 0 references
            group(Details)
            {
                Caption = 'Details', Comment = 'ESP="Detalles"';
                0 references
                field(Cost; Rec.Cost) ···
                0 references
                field(CurrencyCode; Rec.CurrencyCode) ···
                0 references
                field(InvoiceDate; Rec.InvoiceDate) ···
                0 references
                field(Project; Rec.Project) ···
                0 references
                field(Task; Rec.Task) ···
                0 references
                field(Account; Rec.AccountNumber) ···
                0 references
                field(Image; Rec.Image) ···
            }

            You, last month | 1 author (You) | 0 references
            group(BalAccountOptions)
            {
                Caption = 'Bal. Account', Comment = 'ESP="Cuenta de contrapartida"';
                0 references
                field(BalAccountType; Rec.BalAccountType) ···
                0 references
                field(BalAccount; Rec.BalAccount) ···

            }
```

Figura 5.5: Tarjeta de un gasto

Una vez creadas la tabla de datos y las páginas que los muestran, llega la parte de desarrollar acciones que nos permitan registrar los gastos. Todas estas funciones están encapsuladas en *codeunits*. Se han desarrollado dos codeunits, la primera, que es la que se observa en la figura 5.6, realiza inserciones en las tablas, ya creadas por Microsoft, *Gen. Journal Batch*, *No. Series*, *No. Series Line*. En la primera tabla se inserta la información para crear una sección del diario general de proyectos, donde se registrarán los gastos, específica para este tipo de gastos. Las otras dos inserciones en la tabla se ocupan de añadir un número de serie que utilizaremos para nombrar nuestros gastos. Las inserciones siempre se realizan sobre una variable de tipo *Record*, que se inicializa indicando el nombre de la tabla. Para insertar registros se utiliza la función **Init()**, la cual se encarga de inicializar un registro en la tabla; se indican los campos a rellenar y con qué datos y una vez finalizado se inserta con **Insert()**. Esta codeunit tiene una propiedad *Subtype* a la cual le indicamos que es de tipo *Install*; gracias a esto, nos deja implementar un disparador **OnInstallAppPerCompany()** que es donde se realizan las inserciones. Este disparador se activará cada vez que se instale la aplicación, para que así se comience con datos que son de interés para nuestra solución.

```
codeunit 50101 SetUpInstallation
{
    Subtype = Install;
    0 references
    trigger OnInstallAppPerCompany()
    var
        GenJournalBatchRecord: Record "Gen. Journal Batch";
        NoSeriesRecord: Record "No. Series";
        NoSeriesLineRecord: Record "No. Series Line";
    begin
        if not (GenJournalBatchRecord.Get('PROYECTOS', 'GASTOS')) then begin
            GenJournalBatchRecord.Init();
            GenJournalBatchRecord."Journal Template Name" := 'PROYECTOS';
            GenJournalBatchRecord.Name := 'GASTOS';
            GenJournalBatchRecord.Description := 'Diario para los gastos corrientes';
            GenJournalBatchRecord."Copy VAT Setup to Jnl. Lines" := true;
            GenJournalBatchRecord.Insert();
        end;
        if not (NoSeriesRecord.Get('GCOR')) then begin
            NoSeriesRecord.Init();
            NoSeriesRecord.Code := 'GCOR';
            NoSeriesRecord.Description := 'Gastos corrientes';
            NoSeriesRecord."Default Nos." := true;
            NoSeriesRecord."Manual Nos." := true;
            NoSeriesRecord.Insert();
            NoSeriesLineRecord.Init();       You, 4 weeks ago • Now NoSeries are genra
            NoSeriesLineRecord."Series Code" := 'GCOR';
            NoSeriesLineRecord."Starting No." := 'GC00001';
            NoSeriesLineRecord."Ending No." := 'GC99999';
            NoSeriesLineRecord."Increment-by No." := 1;
            NoSeriesLineRecord.Insert();
        end;
    end;
}
```

Figura 5.6: Codeunit de instalación

En la otra *codeunit* se han realizado varias funciones. La primera y la segunda que se observan en la imagen 5.7, consisten en inserciones a las cuales se les pasa el registro actual de la tabla para que se puedan realizar. La primera función se encarga de registrar el gasto en el diario general de proyectos, en la sección "Gastos". Esta sección es la que insertamos en la figura 5.6. La función **Validate()** se encarga de llamar a un disparador **OnValidate()** que se ubica en el campo de la tabla. Este disparador se llama para que valide el dato y ejecute el contenido para una correcta inserción de los datos. La segunda función se encarga de insertar el documento en el proyecto correspondiente. Como la imagen se almacena de tipo *Blob*, se ha tenido que crear una variable *InStream* donde almacenar la lectura de la imagen. Esto se realiza mediante la función **CreateInStream(*InStream*)**. Una vez inicializado el registro en la tabla, se realiza la función **ImportFromStream(*InStream, Name*)**, que se encarga de importar el objeto respuesta, importando así la imagen en un archivo con el nombre que le indiquemos.

```
codeunit 50100 CostCodeunit
    procedure RegisterCost(var CostRecord: Record "Cost")
    var
        GenJournalLineRecord: Record "Gen. Journal Line";
    begin

        GenJournalLineRecord.Init();
        GenJournalLineRecord."Journal Template Name" := 'PROYECTOS';
        GenJournalLineRecord."Journal Batch Name" := 'GASTOS';
        GenJournalLineRecord.Validate("Journal Batch Name");
        GenJournalLineRecord."Line No." := GenJournalLineRecord.GetNewLineNo(GenJournalLin
        GenJournalLineRecord."Posting Date" := WorkDate();
        GenJournalLineRecord.Validate("Posting Date");
        GenJournalLineRecord."Account Type" := "Gen. Journal Account Type"::"G/L Account";
        GenJournalLineRecord."Account No." := CostRecord.AccountNumber;
        GenJournalLineRecord."Document No." := CostRecord.No;
        GenJournalLineRecord.Description := CostRecord.Title;
        GenJournalLineRecord."Bal. Account Type" := CostRecord.BalAccountType;
        GenJournalLineRecord."Bal. Account No." := CostRecord.BalAccount;
        GenJournalLineRecord."Debit Amount" := CostRecord.Cost;
        GenJournalLineRecord.Validate("Debit Amount");
        GenJournalLineRecord."Job Line Type" := "Job Line Type"::Budget;
        GenJournalLineRecord."Job Quantity" := 1;
        GenJournalLineRecord.Quantity := 1;
        GenJournalLineRecord."Job No." := CostRecord.Project;
        GenJournalLineRecord."Job Task No." := CostRecord.Task;
        GenJournalLineRecord.Validate("Job Task No.");
        GenJournalLineRecord."Source Code" := 'DIRGLPROY';
        GenJournalLineRecord."Currency Code" := CostRecord.CurrencyCode;
        GenJournalLineRecord.Validate("Currency Code");
        GenJournalLineRecord."Document Date" := CostRecord.InvoiceDate;
        GenJournalLineRecord.Insert(true, false);

    end;
```

```
procedure AttachDocument(var CostRecord: Record "Cost");
var
    DocumentAttachmentRecord: Record "Document Attachment";
    DocumentInStream: InStream;
    IncomingFileName: Text[250];
begin
    IncomingFileName := CostRecord.No + '_' + CostRecord.Task + '_invoice';
    CostRecord.Image.CreateInStream(DocumentInStream);
    DocumentAttachmentRecord.Init();
    DocumentAttachmentRecord."Table ID" := 167;
    DocumentAttachmentRecord."No." := CostRecord.Project;
    DocumentAttachmentRecord."Attached Date" := CurrentDateTime;
    DocumentAttachmentRecord."File Name" := IncomingFileName;
    DocumentAttachmentRecord."File Type" := "Document Attachment File Type"::Image;
    DocumentAttachmentRecord."File Extension" := 'jpg';
    DocumentAttachmentRecord."Attached By" := UserSecurityId();
    DocumentAttachmentRecord.ImportFromStream(DocumentInStream, IncomingFileName);
    DocumentAttachmentRecord.Insert();
end;
```

Figura 5.7: Funciones de inserción

Las otras dos funciones realizadas se encargan de comprobar que el nombre del documento no se repita, ya que hay que hacer comprobaciones en tablas donde el nombre del documento no forma parte de ninguna clave, y la otra función se encarga de poder descargar el documento.

Respecto a la primera función, realizamos filtros a partir de la tabla que seleccionamos mediante la función **SetFilter()**, filtrando en las tablas que nos interesa por el número del registro a insertar y así comprobar que este no había sido anteriormente registrado. Mediante un **Count()** se comprueba si ya existe o no, e imprimimos los mensajes que se ven en la figura 4.16. En la imagen 5.8 se puede observar el código de la función explicada.

```
procedure NoRepeatDocument(var CostRecord: Record "Cost"; MultiLine: Boolean) multiMessage: Boolean
var
    JobPlanningLineRecord: Record "Job Planning Line";
    GenJournalLineRecord: Record "Gen. Journal Line";
    ResponseAcceptLbl: Label 'Document accepted in the job journal', Comment = 'ESP="El documento se han insertado en el diario de proyectos."';
    ResponseErrorLbl: Label 'Document repeated in the project or journal', Comment = 'ESP="El documento no se ha insertado, ya existe en una tarea o en el diario."';
begin
    JobPlanningLineRecord.SetFilter(JobPlanningLineRecord."Document No.", CostRecord.No);
    GenJournalLineRecord.SetFilter(GenJournalLineRecord."Document No.", CostRecord.No);
    if (MultiLine = false) then
        if ((JobPlanningLineRecord.Count >= 1) or (GenJournalLineRecord.Count >= 1))
        then
            Error(ResponseErrorLbl)
        else
            Message(ResponseAcceptLbl)
    else
        if ((JobPlanningLineRecord.Count >= 1) or (GenJournalLineRecord.Count >= 1))
        then
            multiMessage := false
        else
            multiMessage := true;
end;
```

Figura 5.8: Función NoRepeatDocument

La segunda función se encarga de almacenar en una variable *InStream* el valor de la imagen y, posteriormente, descargarla en el dispositivo mediante la función **DownloadFromStream()**, indicando el nombre con el que se descargará. Previamente, como la imagen ya estaba insertada, ha sido necesario ejecutar la función **CalcFields()** para calcular el valor *Blob* de la imagen. En la figura 5.9 se observa la función desarrollada.

```
procedure DownloadFile(var CostRecord: Record "Cost") download: Boolean
var
    ResponseInStream: InStream;
    tempfilename: Text;
    ErrorAtachmentLbl: Label 'No file avaliable', Comment = 'ESP="No hay ningún archivo disponible"';

begin
    tempfilename := CostRecord.No + '_invoice.jpg';
    CostRecord.CalcFields(Image);
    if not CostRecord.Image.HasValue then
        Error(ErrorAtachmentLbl);
    CostRecord.Image.CreateInStream(ResponseInStream);
    download := DownloadFromStream(ResponseInStream, '', '', '', tempfilename);
end;
```

Figura 5.9: Función DownloadFile

Una vez declaradas estas funciones, creamos acciones para ambas páginas para poder ejecutarlas. La primera acción, llamada DownloadInvoice, tan solo se encarga de ejecutar la función del mismo nombre que se ve en la anterior figura. La llamada a este método debe ponerse en un disparador **OnAction()**, que se ejecutará cada vez que se presione el botón de la acción.

La segunda función, Register, es la encargada de llamar al método RegisterCost, mostrado en la figura anterior, y al método NoRepeatDocument, que se muestra en la imagen anterior. Una vez llamados, si el flujo sigue siendo el correcto, cambia el campo Locked del gasto a verdadero y actualiza la página. Ambas acciones se pueden ver en la figura 5.10.

```
area(Processing)
{
    0 references
    action(DownloadInvoice)
    {
        Caption = 'Download Invoice', Comment = 'ESP="Descargar factura"';
        ToolTip = 'Download Invoice', Comment = 'ESP="Descargar factura"';
        ApplicationArea = All;
        Image = Export;
        0 references
        trigger OnAction();
        begin
            CostCodeunit.DownloadFile(Rec);
        end;
    }
    0 references
    action(Register)
    {
        Caption = 'Register Cost', Comment = 'ESP="Registrar gastos"';
        ToolTip = 'Register', Comment = 'ESP="Registrar gastos"';
        ApplicationArea = All;
        Image = Register;
        0 references
        trigger OnAction()      You, 4 weeks ago • Modify Message
        var
            CostCard: Page CostCard;
        begin
            CostCodeunit.NoRepeatDocument(Rec, false);
            CostCodeunit.RegisterCost(Rec);
            Rec.Locked := true;
            CostCard.SetRecord(Rec);
            CurrPage.Close();
            CostCard.Run();
        end;
    }
}
```

Figura 5.10: Acciones Register y DownloadFile

La última acción añadida se encarga de una redirección a la sección de gastos del diario general de proyectos. En la propiedad *RunObject* de la acción se indica la página a la que se redirigirá, y en la propiedad *RunPageLink* se especifican los filtros necesarios. En la figura, también se observa la inicialización de la codeunit que ejecuta las funciones anteriormente descritas y la definición de un disparador **OnOpenPage()** que se encarga de comprobar cada vez que se abra la página el estado del campo *Locked*. Si este es verdadero, la página pasa a ser no editable. También se ha definido un disparador **OnDeleteRecord()** que, al borrar un registro, comprueba si este está previamente registrado o no, teniendo en cuenta el campo *Registered*. En la figura 5.11 se puede observar tanto la acción como los disparadores definidos.

```
actions
    area(Navigation)
        0 references
        action("OpenJobJournal")
        {
            ApplicationArea = Jobs;
            Caption = 'OpenJobJouurnal', Comment = 'ESP="Diario general de proyectos"';
            Image = Journals;
            RunObject = Page "Job G/L Journal";
            RunPageLink = "Journal Template Name" = filter('PROYECTOS'),
                          "Journal Batch Name" = filter('GASTOS');

            ToolTip = 'Open the job journal, for example, to post usage for a job.', Comment = 'ESP="Abre diario general de proyectos"';

        }
    }
}
You, last month | 1 author (You)
var
    3 references
    CostCodeunit: Codeunit CostCodeunit;
  You, last month • Imputacion gastos APP inserted
0 references
trigger OnOpenPage()
var
    JobPlanningLineRecord: Record "Job Planning Line";
    GenJournalLineRecord: Record "Gen. Journal Line";
begin
    JobPlanningLineRecord.SetFilter(JobPlanningLineRecord."Document No.", Rec.No);
    GenJournalLineRecord.SetFilter(GenJournalLineRecord."Document No.", Rec.No);
    if Rec.Locked then
        CurrPage.Editable(false)
end;
```

Figura 5.11: Acción OpenJobJournal y disparadores

La tabla donde se registra el gasto es *Gen Journal Line*. Hemos creado un objeto de extensión de tabla que se encarga de ejecutar dos disparadores. Uno es **OnAfterDelete()**, que se activa una vez que se borra un gasto del diario y cambia los campos *Registered* y *Locked* a falso. El otro es **OnInsert()**, que se activa cuando se inserta un gasto en el diario y modifica los dos campos anteriores a verdadero. Esto se puede apreciar en la imagen 5.12.

```
tableextension 50101 GenJorunalExt extends "Gen. Journal Line"
{
    1 reference
    trigger OnAfterDelete()
    var
        JobPlanningLineRecord: Record "Job Planning Line";
        CostRecord: Record Cost;
    begin
        JobPlanningLineRecord.SetRange(JobPlanningLineRecord."Document No.", Rec."Document No.");
        if JobPlanningLineRecord.Count() = 0 then begin
            CostRecord.SetRange(CostRecord.No, Rec."Document No.");
            CostRecord.ModifyAll(CostRecord.Registered, false);
            CostRecord.ModifyAll(CostRecord.Locked, false);
        end;

    end;

    1 reference
    trigger OnInsert()
    var
        CostRecord: Record Cost;
    begin
        CostRecord.SetRange(CostRecord.No, Rec."Document No.");
        CostRecord.ModifyAll(CostRecord.Registered, true);
        CostRecord.ModifyAll(CostRecord.Locked, true);
    end;

}       You, last month • Imputación gastos APP inserted
```

Figura 5.12: TableExtension de la tabla del diario de proyectos

También se ha creado un objeto *PageExtension* que extiende la funcionalidad de registro de la página Job G/L Journal, la cual ya estaba en el sistema y corresponde al diario general de proyectos. Una vez que los gastos están almacenados aquí, se registran para generar movimientos tanto contables como en el proyecto.

Se ha creado una extensión de esta página para gestionar la adjunción del documento al proyecto correspondiente, ya que éste debe registrarse tan solo cuando registramos el gasto desde el diario. Para lograr esto, he creado un disparador **OnBeforeAction()** que, antes de ejecutar la acción *Post*, almacena en una lista los gastos actuales que hay en el diario y la sección correspondiente, y guarda el número de registros que hay en la tabla Job Planning Line, donde se almacenan los datos de todos los proyectos.

Una vez que tenemos la lista, se ejecuta el disparador **OnAfterAction()** que itera la lista, busca el gasto en la tabla *Cost*, y para cada gasto, calcula el valor de la imagen y lo inserta en el proyecto correspondiente mediante la función **AttachDocument()** que se observa en la figura 5.7. En la imagen 5.13 se muestra la definición de los dos disparadores:

```
pageextension 50101 JobGLExtension extends "Job G/L Journal"
    actions
        modify("P&ost")
            trigger OnBeforeAction()
            var
                GenJorunalLineRecord: Record "Gen. Journal Line";
                JobPlanningLine: Record "Job Planning Line";
            begin
                l1.RemoveRange(1, l1.Count());
                if (Rec."Journal Template Name" = 'PROYECTOS') and (Rec."Journal Batch Name" = 'GASTOS') then begin
                    GenJorunalLineRecord.SetRange("Journal Template Name", 'PROYECTOS');
                    GenJorunalLineRecord.SetRange("Journal Batch Name", 'GASTOS');
                    journalcount := JobPlanningLine.Count();
                    if GenJorunalLineRecord.Find('-') then
                        repeat
                            l1.Add(GenJorunalLineRecord."Document No.");
                        until GenJorunalLineRecord.Next() = 0;
                    end;
                end;
                1 reference
                trigger OnAfterAction()
                var
                    CostRecord: Record Cost;
                    GenJorunalLineRecord: Record "Gen. Journal Line";
                    JobPlanningLine: Record "Job Planning Line";
                    CostCodeunit: Codeunit CostCodeunit;
                    iterator: Integer;
                begin
                    if (Rec."Journal Template Name" = 'PROYECTOS') and (Rec."Journal Batch Name" = 'GASTOS') then begin
                        GenJorunalLineRecord.SetRange(GenJorunalLineRecord."Journal Template Name", 'PROYECTOS');
                        GenJorunalLineRecord.SetRange(GenJorunalLineRecord."Journal Batch Name", 'GASTOS');
                        GenJorunalLineRecord.FindFirst();
                        if journalcount <> JobPlanningLine.Count() then begin
                            iterator := 1;
                            repeat
                                CostRecord.SetRange(CostRecord.No, l1.Get(iterator));
                                CostRecord.FindFirst();
                                CostRecord.CalcFields(Image);
                                CostCodeunit.AttachDocument(CostRecord);
                                iterator := iterator + 1;
                            until iterator > l1.Count();
                            GenJorunalLineRecord.DeleteAll();
                        end;
                    end;
                end;
            }
```

Figura 5.13: PageExtension de la página del diario de proyectos

Para pasar datos de Business Central a Power Apps, se han creado páginas de tipo API. Estas páginas permiten generar puntos de acceso a servicios web. Se han generado dos páginas: una con los datos de nuestra tabla y otra con los datos de los proyectos. Business Central por defecto ya tiene páginas definidas de este tipo, pero para los proyectos no hay ninguna. La estructura de estas páginas se puede ver en la figura 5.14, la cual corresponde a la tabla de gastos.

```
page 50102 CostAPI
{
    PageType = API;
    Caption = 'ZYBookApi';
    APIPublisher = 'zy';
    APIGroup = 'powerapps';
    APIVersion = 'v2.0';
    EntityName = 'cost';
    EntitySetName = 'cost';
    ODataKeyFields = SystemId;
    SourceTable = Cost;
    DelayedInsert = true;
    DeleteAllowed = true;
    ModifyAllowed = true;

    You, last month | 1 author (You)
    layout
    {
        You, last month | 1 author (You) | 0 references
        area(Content)
        {
            repeater(GroupName)
            {
                0 references
                field("system"; Rec.SystemId)...
                0 references
                field(description; Rec.Description)...
                0 references
                field(cost; Rec.Cost)...
                0 references
                field(image; Rec.Image)...
                0 references
                field(project; Rec.Project)...
                0 references
                field(task; Rec.Task)...
                0 references
                field(title; Rec.Title)...
                0 references
                field(email; Rec.Email)...
                0 references
                field(invoiceDate; Rec.InvoiceDate)...
                0 references
                field(currencyCode; Rec.CurrencyCode)...
                0 references
                field(number; Rec.No)...
            }
        }        You, last month • Imputación gastos APP
    }
```

Figura 5.14: Página API de la tabla gastos

Una vez terminada la parte de Business Central, pasamos a la de Power Apps. Lo primero que se realiza en la app es configurar la tabla de gastos como origen de datos mediante la página API creada anteriormente. Una vez hecho esto, para que la lista de gastos muestre únicamente los gastos del usuario, se ha añadido al disparador **OnStart()** de la app la instrucción de generar una variable *emailTable* cada vez que se inicie la aplicación. Esta variable almacena la lista de los gastos según el email del usuario. También se inicializan a falso las variables que gestionan las ventanas emergentes. En la imagen 5.15 se muestra cómo se ha definido este código.

```
Set(emailTable; Search([@'cost,zy%252Fpowerapps%252Fv2.0'];User().Email;email));;
Set(showGroup;false);;
Set(showIncorret;false)
```

Figura 5.15: Disparador OnStart()

Para mostrar la lista de gastos en la propiedad *Item* de la galería, se ha utilizado la función **SortByColumns()** que primero muestra los gastos si hay una coincidencia en el texto buscado con algún concepto o proyecto de la lista. Si el cuadro está vacío, se muestran todos los gastos. Luego, este resultado se ordena por la variable *invoiceDate*. Dependiendo del resultado de la búsqueda y el valor actual de la variable *SortingDescending1*, se ordenará de más antiguo a más nuevo o viceversa. En la figura 5.16 se puede observar la utilización de la función **SortByColumns()**.

```
SortByColumns(Search(emailTable; TextSearchBox1.Text; title; project); "invoiceDate"; If(SortDescending1; SortOrder.Descending; SortOrder.Ascending))
```

Figura 5.16: Condiciones para mostrar gastos

La inserción de datos se lleva a cabo mediante la ejecución de un flujo de datos configurado en Power Automate. Para invocar este flujo, primero debemos crearlo. En la imagen 5.17 se muestran los datos que Power Automate recibirá de Power Apps y sus tipos de variables.

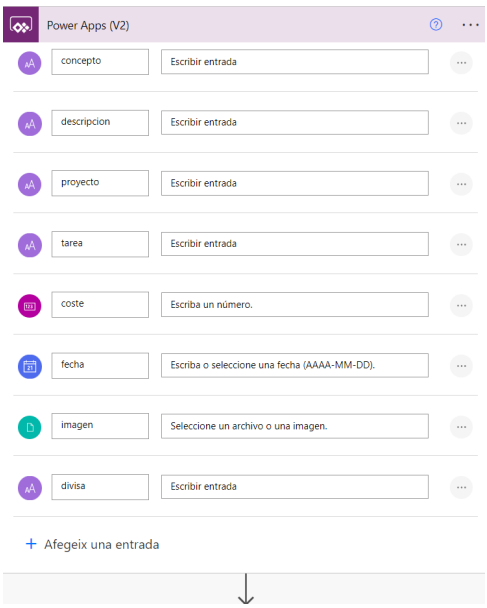


Figura 5.17: Datos recibidos de Power Apps

Una vez Power Automate recoge los datos, continúa con el flujo insertando en el entorno,

compañía y tabla seleccionados, emparejando las variables que recoge Power Automate con las variables de la tabla gastos según su página API. Una vez insertado, para poder adjuntar la foto, se actualiza el registro anteriormente creado mediante un identificador del sistema que se obtiene tras la inserción. En la figura 5.18 se observa cómo se definen estos pasos.

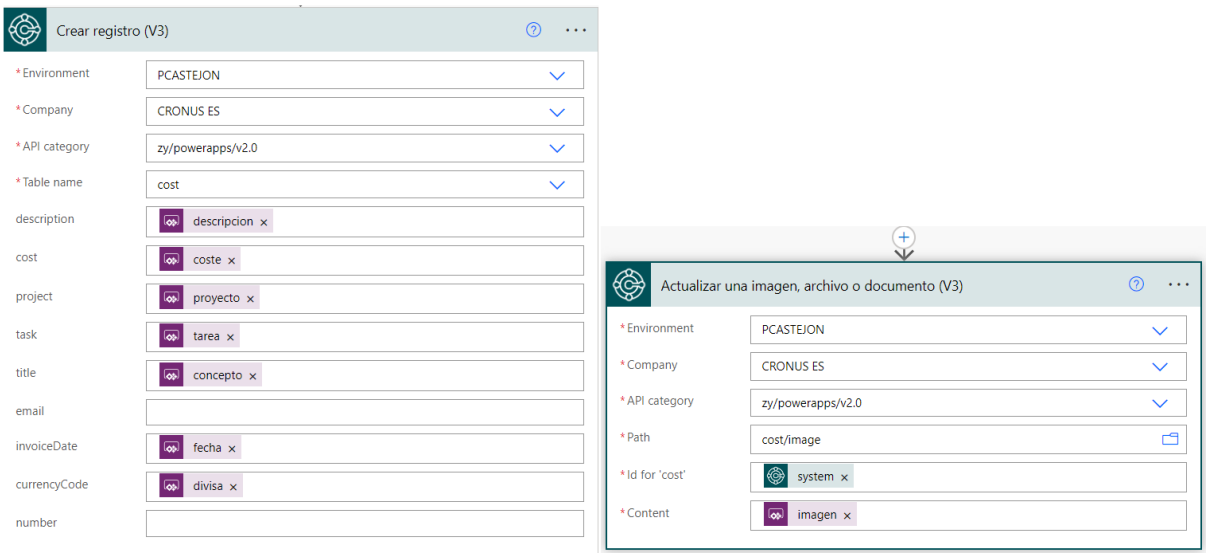


Figura 5.18: Inserción en Business Central

Una vez definido el flujo en Power Automate, desde el botón de insertar, el que se observa en la figura 4.8 en el disparador **OnSelect()** llama al código que se observa en la imagen 5.19.

```
If(IsBlank(CurrencyDropdown.SelectedText.Value); Set(currencyCode;""); Set(currencyCode; CurrencyDropdown.SelectedText.Value));;
If(IsBlank(TitleValue.Text) || IsBlank(DescriptionValue.Text) ||  IsBlank(CostValue.Text) || UploadedImage1.Image = SampleImage;
//True
Set(showIncorret; true);
//False
Insert.Run(TitleValue.Text;DescriptionValue.Text;ProjectDropddown.SelectedText.Value;TaskDropwdown.SelectedText.Value;CostValue.Text;
Text(InvoiceDateValue.SelectedDate; "yyyy-mm-dd");{text_4: currencyCode; file: {name: TaskDropwdown.SelectedText.Value & ".jpg"; contentBytes: UploadedImage1.Image}}));;
If(showIncorret = false;Set(showGroup;true))
```

Figura 5.19: Código llamado al insertar

Este código se encarga de comprobar que, si el valor del código de divisa a insertar está vacío, genera una variable con un texto vacío, sino genera la variable con el valor seleccionado. Esto se debe a que el flujo en Power Automate no puede recibir nulos. Después ejecuta una condición que es la que se encarga de ejecutar o no el flujo de datos. Se comprueba que los datos del formulario estén sin rellenar, si es así, se muestra la ventana emergente de error. Si no, se llama al flujo de datos llamado *Insert* y se le pasan los datos que veíamos en la figura 5.17. Si se llega a insertar, mostrará por pantalla un mensaje avisando de que la inserción se ha realizado.

5.3. Verificación y validación

Para comprobar el funcionamiento de las funciones principales, se han desarrollado varios test que comprueban que todo se ejecuta como debería. Para realizar esto, en Business Central se ha creado un proyecto aparte. A este proyecto le añadimos una *codeunit* y en la propiedad *Subtype* le indicamos que es de tipo Test. Esto último ayudará a que Business Central detecte esta *codeunit* en su herramienta de test para poder ejecutarlos.

Cada prueba se identifica por su cabecera [Test] y se define mediante el patrón 'Given-When-Then' [3] . En la imagen 5.20 se observa la definición del test que se encarga de adjuntar el documento. Para esto se inserta primero un gasto de prueba y le insertamos una imagen del sistema, la cual actuará como ticket. Una vez insertado, cuando se quiere adjuntar el documento en el proyecto, se comprueba que la inserción se ha realizado correctamente.

```
[Test]
0 references
procedure AttachDocumentTest()
var
    CostRecord: Record Cost;
    MediaRecord: Record "Tenant Media";
    DocumentAttachmentRecord: Record "Document Attachment";
    CostCodeunit: Codeunit CostCodeunit;
    FileName: Text;

begin
    // [Scenario]

    // [Given] Setup:
    CostRecord.Init();
    CostRecord.No := 'PRUEBATEST';
    CostRecord.Title := 'Test';
    CostRecord.Description := 'Test';
    CostRecord.Cost := 150;
    CostRecord.InvoiceDate := 20240215D;
    CostRecord.Project := 'P00020';
    CostRecord.Task := 'TEST';
    CostRecord.Insert();
    MediaRecord.FindFirst();
    MediaRecord.CalcFields(Content);
    CostRecord.FindLast();
    CostRecord.Image := MediaRecord.Content;
    CostRecord.Modify(false);
    FileName := CostRecord.No + '_' + CostRecord.Task + '_invoice';
    // [When] Exercise:
    CostCodeunit.AttachDocument(CostRecord);
    // [Then] Verify:
    DocumentAttachmentRecord.SetRange("File Name", FileName);
    ExpectedValue := 1;
    ActualValue := DocumentAttachmentRecord.Count();
    IfErrorTxt := 'El documento no se registró';
    AssertThat.AreEqual(ExpectedValue, ActualValue, IfErrorTxt);
end;
```

Figura 5.20: Test de AttachDocument

Aparte del test observado en la figura 5.20 se han realizado pruebas para la función de descargar factura, donde se comprueba que la función *DownloadFile* llega a descargar la factura, para la función de registrar en el diario, la cual comprueba que la inserción en el Diario general de proyectos se ha realizado, buscando en la tabla correspondiente si se encuentra el gasto registrado. Por último, se ha comprobado que la inserción de gastos se realiza de manera correcta. Si el formulario no se rellena al completo, a excepción del código de divisa, saltará un error, sino la inserción ocurrirá. Esta última prueba se ejecuta desde Power Automate comprobando que el flujo de la figura 5.18 se ejecute correctamente. En la imagen 5.21 se puede observar cómo la prueba se ha realizado, ha sido satisfactoria y muestra el tiempo empleado por cada acción del flujo.

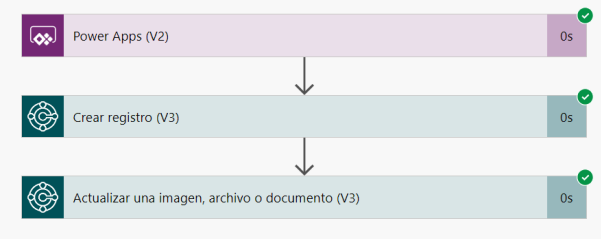


Figura 5.21: Prueba del flujo en Power Automate

Una vez realizados los distintos test, estos se pueden ejecutar de manera local en Business Central. El ERP tiene una extensión llamada AL Test Tool que se encarga de ejecutar los test. Para esto hay que subir el fichero .app que genera el proyecto de test al compilar. Una vez subidos los test desde la extensión AL Test Tool, detectamos los codeunits de tipo Test, añadiendo las funciones que se testan. Podemos personalizar qué test poner y cuáles no y ejecutarlos, mostrándonos por pantalla si se han pasado y cuánto han tardado. En la figura 5.22 se muestran los resultados de las pruebas que se han desarrollado.

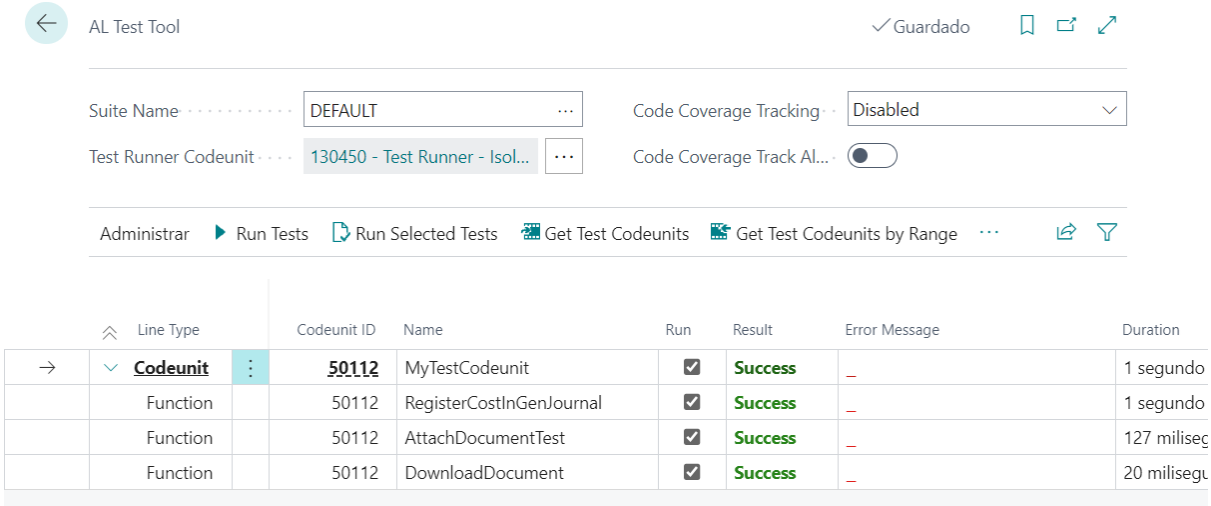


Figura 5.22: Ejecución de los test en la extensión AL Test Tool

Por último, al estar el proyecto alojado en Github, en específico en una plantilla creada por Microsoft llamada AL-Go, y estar en un entorno de integración continua/entrega o implementación continua cada vez que se realice un commit, el entorno se encargará de crear un entorno de pruebas de Business Central y ejecutar los test. En la imagen 5.23 se muestra la ejecución de las pruebas en el repositorio de Github.

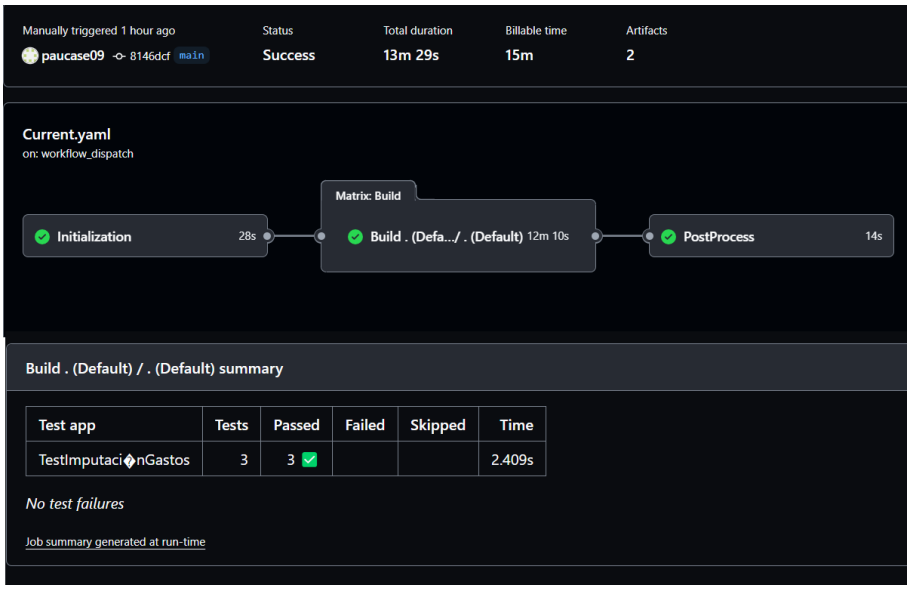


Figura 5.23: Resultado tests en Github

Capítulo 6

Conclusiones

Para finalizar la memoria se van a comentar aspectos como los resultados alcanzados, conocimientos adquiridos, posible trabajo futuro respecto al proyecto y para finalizar conclusiones personales.

A nivel tanto de proyecto como de producto, el resultado es satisfactorio, cumpliendo todos los objetivos que fueron planteados al principio de esta memoria. Una vez se lograron los objetivos, se habían propuesto mejoras a nivel funcionalidad, como el hecho de poder registrar varias líneas a la vez, y un campo que identificara qué había sido registrado. Se plantearon otras mejoras, como una aprobación del documento por parte de un superior, esto quiere decir que las modificaciones hechas por un contable deban ser previamente aceptadas por un administrador empresarial. O que se pueda aceptar, registrar e insertar el gasto con tan solo un clic. Esta opción sería viable para empresas donde un mismo trabajador tiene varios roles y no en otras algo más grandes que cada uno tiene su rol. Estas dos últimas mejoras no se llegaron a desarrollar y se han propuesto como mejoras futuras.

A nivel formativo, básicamente se ha aprendido todo lo utilizado. Cuando entré, tenía una definición muy pobre de lo que era un ERP. Tras la realización de la formación, se han adquirido desde conceptos técnicos a prácticos. Ahora conozco en mayor medida lo que es un ERP y los que hay en el mercado y más en concreto en el que hemos trabajado, Microsoft Dynamics Business Central. En la parte práctica, he aprendido a utilizar herramientas que nunca usé antes, como pueden ser Power Apps y Power Automate. También se aprendió a programar desde cero y conocer el lenguaje AL para poder desarrollar, de ahora en adelante, extensiones para el ERP.

En el ámbito profesional hemos aprendido a tener reuniones con el cliente. Una vez terminada la formación tanto funcional como técnica, tuvimos una reunión con nuestro supervisor, el que simuló ser un cliente para sacar más información sobre el producto a desarrollar. Además de realizar una presentación sobre la nueva funcionalidad y recibir comentarios sobre cómo mejorarla.

Personalmente, pienso que este ha sido un proyecto ambicioso, en el que he adquirido muchos conocimientos. Además, he tenido una evolución personal a lo largo del proceso, ya que entrar en una empresa con gente desconocida ha hecho que dejara atrás mis inseguridades para

desenvolverme en un entorno nuevo. Por último, tras la finalización de la estancia, se me ofreció seguir en la empresa. De hecho, escribo estas últimas líneas desde la oficina, mientras espero que me asignen un proyecto. Estoy ilusionado con esta nueva etapa que se ha dado gracias a la estancia en prácticas.

Bibliografía

[1] IBM. ¿qué es la arquitectura de tres niveles? https://www.ibm.com/mx-es/topics/three-tier-architecture. [Consulta: 1 de Abril de 2024].

[2] Mario Arias Escalona. Guia de licenciamiento de power platform. https://www.thepowerplatformcave.com/guia-licenciamiento-powerplatform-2023/. [Consulta: 15 de Marzo de 2024].

[3] Martin Fowler. Given when then. https://martinfowler.com/bliki/GivenWhenThen.html. [Consulta: 20 de Marzo de 2024].

[4] Microsoft. Business central component and system topology. https://learn.microsoft.com/en-us/dynamics365/business-central/dev-itpro/deployment/product-and-architecture-overview. [Consulta: 4 de Abril de 2024].

[5] Microsoft. Power platform. https://www.microsoft.com/es-es/power-platform. [Consulta: 4 de Abril de 2024].

[6] Microsoft. Precios de dynamics 365 business central. https://dynamics.microsoft.com/es-es/business-central/pricing/. [Consulta: 15 de Marzo de 2024].

[7] Microsoft. ¿qué es sql server? https://learn.microsoft.com/es-es/sql/sql-server/what-is-sql-server?view=sql-server-ver16. [Consulta: 8 de Abril de 2024].