Gestión de una tienda online con Odoo

Jan Tugores Castells

Resumen– Este proyecto se enfoca en la implementación de Odoo, versión 15, para gestionar una tienda en línea, abarcando aspectos clave como la contabilidad y la integración de una API que facilita el circuito de compras y ventas. Con el objetivo de optimizar los procesos, se ha utilizado la metodología ágil por sprints para un desarrollo eficiente. En este informe final, se presentarán los resultados obtenidos, destacando los logros alcanzados y los desafíos superados en cada etapa. A lo largo de este informe, se detallarán los aspectos más relevantes del proyecto, brindando una visión global de las funcionalidades implementadas y su impacto en la gestión de la tienda online.

Palabras clave– Odoo, Python, tienda online, gestión integral, contabilidad, automatización, API personalizada, circuito de compras y ventas, sprints ágiles, eficiencia operativa.

Abstract– This project focuses on the implementation of Odoo, version 15, for managing an online store, covering key aspects such as accounting and the integration of an API that facilitates the purchasing and sales process. With the aim of optimizing processes, an agile methodology using sprints has been employed for efficient development. This final report will present the obtained results, highlighting the achieved milestones and the challenges overcome at each stage. Throughout this report, the most relevant aspects of the project will be detailed, providing a comprehensive overview of the implemented functionalities and their impact on the management of the online store.

Keywords– Odoo, Python, online store, comprehensive management, accounting, automation, custom API, purchasing and sales process, agile sprints, operational efficiency

1 Introducción - Contexto del trabajo

Generalmente, en los negocios, hay varios temas a gestionar y no siempre están centralizados. A centralizados me refiero a que no utilizan la misma herramienta para todo. Quizás para la contabilidad utilizan un programa en concreto, para llevar las fichas de los clientes con sus suscripciones, utilizan otro, para gestionar dispositivos de las instalaciones... Entonces esto provoca que la eficiencia de los trabajadores sea muy limitada y que quede expuesta la integridad de datos. Al final, la mayoría de estos procesos dependen de los demás y para que los datos estén actualizados a tiempo real es necesario que todos los sistemas estén integrados entre sí.

• E-mail de contacto: 1494134@uab.cat
• Mención realizada: Enginyeria del Software
• Trabajo tutorizado por: Zakariae Majjouty (Àrea de Ciències de la Computació i Intel·ligència Artificial)
• Curso 2022/23

1.1. Problemática

Llegados a este punto, podemos sacar como conclusión que para mantener todo el sistema funcionando, es necesario tener en cuenta estos dos puntos esenciales:

- **Integraciones:** Como se ha comentado, las integraciones entre las diferentes aplicaciones son necesarias. Por lo general, cada aplicación tiene diferentes campos y datos, lo que hace que la comunicación entre ellas sea muy compleja. La complejidad va aumentando a medida que vamos añadiendo aplicaciones de por medio. Así que podemos asegurar que, a menos aplicaciones, menos complejidad tendremos.
- **Coste:** Este es un punto muy crítico, puesto que, al final, el objetivo principal de cualquier empresa es maximizar los beneficios. Al no tenerlo todo centralizado, nos llegan costes por todos lados. Para empezar, tenemos los costes de las licencias de las aplicaciones, que son generalmente elevados. Luego tenemos los costes de los trabajadores (con sus respectivas formaciones) necesarios para operar en todas las aplicaciones. También se deben tener en cuenta las integraciones, que sobre todo tienen un coste elevado en el momento del desarrollo. Por último, a todo esto, se le suma el coste de mantenimiento total. Todo esto se concentra en la

problemática de no tenerlo todo centralizado perdiendo la oportunidad de maximizar los beneficios.

En el caso de nuestro cliente, su problemática principal es informatizar la gestión contable. El hecho de resolver este problema abre las puertas a que los puntos comentados anteriormente mejoren drásticamente.

1.2. Solución

Para resolver la problemática, se debe buscar una solución que optimice los recursos. Para ello se enfocará la solución en un ERP que cumpla con todos los requisitos necesarios para optimizar la gestión y los gastos del negocio.
Este ERP es Odoo, un software de gestión que concentra todas las aplicaciones en una sola. Odoo es un programa desarrollado sobre una arquitectura web y cada aplicación puede ser instalada desde la propia interfaz según las necesidades del cliente. Al ser de código abierto, se puede encontrar fácilmente el código en GitHub y además cuenta con la OCA (Odoo Community Association) que es dónde las diferentes empresas que desarrollan Odoo pueden aportar mejoras a la comunidad sin ánimo de lucro. Esto es debido a que gracias a que el código está cubierto bajo la licencia LGPLv3 (Lesser General Public License), es posible realizar modificaciones de código y publicarlas respetando siempre los términos de la licencia.
Entonces podemos llegar a la conclusión que por su coste y flexibilidad, la opción más acertada en el caso de nuestro cliente, es utilizar Odoo para la gestión contable de su negocio.

2 OBJETIVOS

El objetivo del proyecto es gestionar la contabilidad de una tienda online, a la par que se gestiona todo el inventario, las compras y las ventas con el operador logístico. De esta manera se le saca mucho partido a Odoo, ya que unifica lo que en principio son dos flujos diferentes en un mismo sistema. Además, fortalece la integridad de los datos y hace que el cliente sienta que tiene un sistema mucho más robusto.
A continuación, el listado de todos los objetivos detallados:

- **Configuración del entorno con Docker**: Se establecerá toda la configuración del entorno utilizando Docker para gestionar cada servicio en contenedores separados, incluyendo Odoo, la base de datos y otros componentes necesarios. Esto permitirá una configuración ágil y facilitará los cambios de configuración.
- **Configuración de la herramienta utilizada para gestionar la base de datos**: Se realizará la configuración adecuada de la herramienta de gestión de la base de datos, considerando las necesidades específicas de la tienda online y la integración con los operadores logísticos. Esto asegurará un almacenamiento eficiente de los datos y permitirá una gestión adecuada de la información.
- **Control de versiones con repositorio de GitHub**: Se implementará un sistema de control de versiones utilizando un repositorio de GitHub para mantener un registro de las versiones del código y del entorno a medida que se desarrolla el proyecto. Esto permitirá un seguimiento preciso de la evolución del proyecto y asegurará la disponibilidad de una versión estable.
- **Configuración inicial de la compañía en Odoo**: Se realizará la configuración inicial de la compañía en Odoo, teniendo en cuenta los aspectos específicos de la tienda online y los operadores logísticos. Esto incluirá la configuración de los datos de la empresa, los detalles de contacto, la configuración fiscal y otros parámetros relevantes.
- **Configuración de la aplicación de Contabilidad**: Se investigará y configurará adecuadamente la aplicación de Contabilidad en Odoo, considerando los requerimientos específicos del negocio, la tienda online y la integración con los operadores logísticos. Esto permitirá una gestión contable, completa y precisa de las transacciones comerciales.
- **Configuración de aplicaciones varias**: A medida que se configure la aplicación de Contabilidad, surgirán otras aplicaciones adicionales que serán relevantes para el negocio en línea y la integración logística. Se instalarán y configurarán los módulos necesarios para cubrir las funcionalidades requeridas, como gestión de inventario, gestión de pedidos, seguimiento de envíos, entre otros.
- **Desarrollar una API integrada con Odoo**: Se desarrollará una API personalizada que permita la comunicación entre Odoo y los sistemas de los operadores logísticos. Esta API facilitará el intercambio de información, permitiendo la creación, actualización y eliminación de registros en la base de datos de Odoo desde los sistemas externos. Además, posibilitará la migración de datos desde aplicaciones antiguas hacia Odoo y mejorará la eficiencia de los procesos logísticos.

Cabe mencionar que se ha optado por eliminar el punto de alojar el proyecto en un servidor web. Se ha hecho un análisis exhaustivo de las diferentes opciones para alojar el proyecto y se ha acabado decidiendo que tener alojado el proyecto en un servidor, no compensa el trabajo que conlleva realizarlo. Es preferible destinar los recursos a otras tareas, que aportan mucho más valor, sobre todo en la fase de preproducción.

3 ESTADO DEL ARTE

El estado del arte en la gestión de tiendas online con Odoo abarca una amplia gama de herramientas y tecnologías que desempeñan un papel fundamental en el desarrollo y funcionamiento de estos sistemas. A continuación, se presenta una revisión detallada de algunas de las herramientas y tecnologías clave utilizadas en el proyecto.

Python: Python es un lenguaje de programación versátil y de alto nivel que se destaca por su legibilidad y facilidad de uso. Es el lenguaje base en el cual se ha desarrollado Odoo, lo que le brinda una base sólida y un amplio ecosistema de bibliotecas y frameworks que facilitan el

desarrollo de aplicaciones web y empresariales.

XML para las vistas: Odoo utiliza XML como lenguaje de marcado para definir las vistas de usuario y las estructuras de datos en su sistema. XML proporciona una forma estructurada y legible de describir la interfaz de usuario y las relaciones entre los diferentes elementos en Odoo.

PostgreSQL: PostgreSQL es un sistema de gestión de bases de datos relacional de código abierto. Odoo utiliza PostgreSQL como su motor de base de datos, lo que garantiza una gestión eficiente y segura de los datos relacionados con la tienda online. Además, Odoo dispone de un ORM (Object Relational Mapping) el cual facilita mucho la comunicación entre el código desarrollado y la base de datos.

API REST: Una API REST (Representational State Transfer) es una interfaz de programación de aplicaciones que permite la comunicación entre diferentes sistemas de software. En el proyecto, se ha utilizado una API REST personalizada para facilitar la integración de servicios externos y permitir la interacción con otros sistemas en el circuito de compras y ventas.

Postman: Postman es una herramienta de colaboración y desarrollo de API que permite realizar pruebas, depurar y documentar APIs. Se ha utilizado Postman para probar y validar la API REST implementada en el proyecto, lo que ha facilitado la verificación y la corrección de posibles errores en la comunicación con otros sistemas.

Swagger: Swagger es un conjunto de herramientas que permite diseñar, construir, documentar y consumir servicios web RESTful. Se ha utilizado Swagger para documentar la API REST personalizada desarrollada en el proyecto, lo que ha facilitado la comprensión y el uso de la API por parte de otros desarrolladores o sistemas externos.

Trello: Trello es una herramienta de gestión de proyectos basada en tableros que permite organizar y realizar un seguimiento de las tareas y actividades del equipo. Se ha utilizado Trello para la planificación y seguimiento de las tareas durante el desarrollo del proyecto, lo que ha facilitado la gestión eficiente del tiempo.

Visual Studio Code: Visual Studio Code es un entorno de desarrollo integrado (IDE) ampliamente utilizado que ofrece una amplia gama de funcionalidades y extensiones para el desarrollo de aplicaciones en diversos lenguajes de programación. Se ha utilizado Visual Studio Code como el entorno de desarrollo principal en el proyecto, lo que ha proporcionado una interfaz intuitiva y herramientas útiles para el desarrollo de código.

4 METODOLOGÍA

Cuando se empieza con el desarrollo de un proyecto, por lo general, de software, es importante que se definan bien las etapas de este y los bloques de desarrollo que va a tener. En el caso de este proyecto, se ha optado por dividir el desarrollo del proyecto en cinco fases siguiendo el estándar que define el PMBOK, siguiendo unas buenas prácticas que nos permita tener un buen seguimiento del proyecto y sus etapas:

1. **Inicial**: Esta fase es para definir los objetivos del proyecto y el alcance que tendrá. Se busca establecer el cómo se desarrollará el proyecto.

2. **Planificación**: Establecer la planificación que se deberá seguir. A partir de aquí, saldrán las demás fases del proyecto ya definidas.

3. **Ejecución**: Esta es la fase que más ocupará de todo el proyecto. En esta nos centraremos en todo el desarrollo del proyecto, desde la configuración inicial del entorno hasta el desarrollo de código. Esta fase estará dividida en distintos bloques con tal de organizar mejor el desarrollo:

    - **Configuración inicial**: Este ERP es muy completo y a su vez muy complejo de configurar, ya que dispone de muchas funcionalidades. Lo que se propone en este bloque es configurar Odoo tal y como viene de base (sin ningún módulo instalado) y una vez hecha esta configuración, configurar los distintos módulos que se utilizarán.
    - **Configuración Odoo**: Este ERP es muy completo y a su vez muy complejo de configurar, ya que dispone de muchas funcionalidades. Lo que se propone en este bloque es configurar Odoo tal y como viene de base (sin ningún módulo instalado) y una vez hecha esta configuración, configurar los distintos módulos que se utilizarán.
    - **Desarrollo**: El bloque de desarrollo sobre todo irá centrado en la API. Aquí se propone desarrollar un módulo de cero, el cual permita recibir llamadas del exterior para crear, actualizar o eliminar registros de la base de datos. Dentro de este bloque también se incluyen pequeñas modificaciones que se tengan que hacer, ya sean visuales o a nivel de modelo. Este bloque puede ser intercalado con el bloque de la configuración de Odoo porque no es necesario que esté configurado el ERP para poder empezar con el desarrollo de código.
    - **Validación**: Este último bloque se va a centrar en validar todos y cada uno de los requisitos del cliente para ver si realmente se cumplen como es debido. Es una fase que además se irá haciendo a medida que se va desarrollando el proyecto.

4. **Monitorización**: Esta fase se va a mantener durante todo el proyecto y será para ir controlando los errores y posibles imprevistos que vayan saliendo.

5. **Cierre**: La fase de cierre se hará pocas semanas antes de la finalización del proyecto y servirá para tener una vista global de todo lo que se ha hecho hasta el momento. En este punto también se incluye el informe final y la presentación del proyecto.

Para el desarrollo de la tercera fase se ha utilizado una metodología Agile. Es una forma ágil de organizar el desarrollo

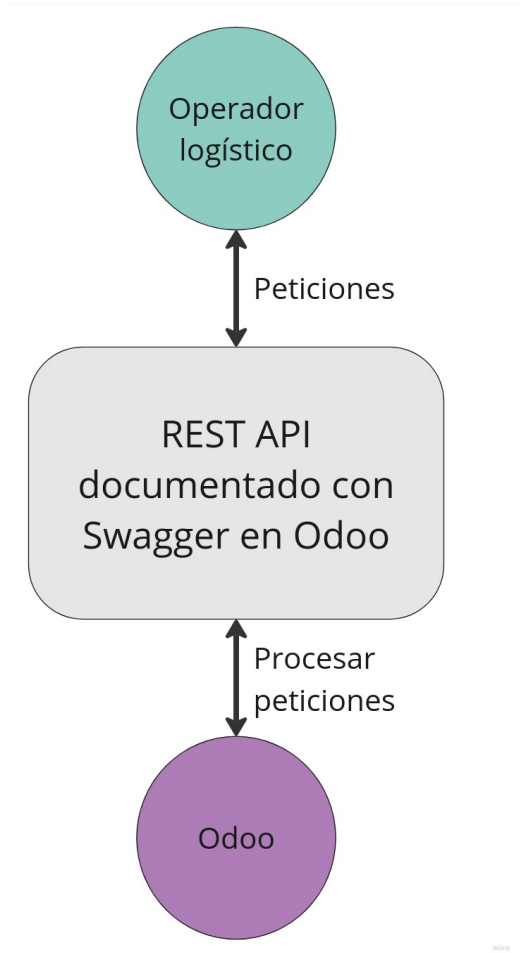

Fig. 1: Comunicación Operador lógistico vs. Odoo

por iteraciones denominadas esprints. La duración de cada esprint ha sido de dos semanas, lo que ha permitido una planificación y ejecución eficiente de las tareas. Al finalizar cada esprint, se ha realizado una revisión de los resultados obtenidos y se ha planificado el siguiente esprint, teniendo en cuenta los objetivos y la planificación del proyecto.
Para la gestión de tareas y seguimiento del proyecto, se ha utilizado la herramienta Trello. A través de Trello, se ha creado un tablero donde se han registrado todas las tareas y su estado de desarrollo.
El tablero de Trello ha estado dividido en columnas que representan las etapas del flujo de trabajo, como ”To-do”, Ïn progress”, ”To validatez ”Done”. Cada tarea ha sido representada como una tarjeta en el tablero, con su descripción detallada.
Gracias a esta metodología de trabajo ágil y al uso de Trello como herramienta de gestión, se ha logrado mantener un seguimiento constante del progreso del proyecto, adaptándose de manera flexible a los cambios.

4.1. Planificación

En la Figura 2 se muestra el diagrama de Gantt correspondiente a la planificación del proyecto.

5 Requisitos

A continuación se detallan los requisitos del proyecto:

TABLA 1: Requisitos Funcionales

ID	Descripción
R1	El sistema debe ser instalado en un entorno de desarrollo o en un servidor local.
R2	El sistema debe permitir la creación de una base de datos para la tienda online, donde se configurarán los parámetros iniciales.
R3	El sistema debe proporcionar la funcionalidad para configurar diferentes niveles de acceso y permisos para los usuarios, como administradores, contadores y empleados del operador logístico.
R4	El sistema debe permitir el registro y seguimiento de transacciones financieras, como ventas, compras, gastos e ingresos.
R5	El sistema debe ofrecer la funcionalidad para administrar el inventario de la tienda online, incluyendo el seguimiento de productos y las existencias.
R6	El sistema debe permitir la integración con el operador logístico para intercambiar información sobre pedidos, envíos y seguimiento de paquetes.
R7	Se deben identificar los requisitos de información y los tipos de datos necesarios para la comunicación entre Odoo y el sistema del operador logístico.
R8	El sistema debe proporcionar endpoints o puntos de acceso en la API para permitir la comunicación bidireccional entre los sistemas.
R9	El sistema debe realizar la validación de datos para asegurar la coherencia y precisión de la información transmitida entre los sistemas.
R10	Se deben llevar a cabo pruebas exhaustivas para verificar el correcto funcionamiento de la API y el intercambio adecuado de información.

6 Casos de uso

Definición de los casos de uso en función de los requisitos funcionales y las fases del proyecto.

6.1. Fase 1: Configuración inicial del entorno

1. **Instalar Odoo**: El administrador del sistema instala Odoo en el servidor o entorno local.
   - *a*) El administrador del sistema configura una infraestructura con Docker para ejecutar Odoo.
   - *b*) El administrador del sistema configura los parámetros de la máquina.
   - *c*) El sistema verifica la instalación exitosa de Odoo.

2. **Configurar parámetros iniciales**: El administrador configura los parámetros iniciales en Odoo.
    a) El administrador accede al panel de administración de Odoo.
    b) El administrador configura los parámetros, como nombre de la empresa, moneda, zona horaria, etc
    c) Los cambios se guardan y aplican en el sistema.
3. **Gestionar usuarios y permisos**: El administrador configura los usuarios y sus permisos en Odoo.
    a) El administrador accede al panel de administración de Odoo.
    b) El administrador crea y configura diferentes tipos de usuarios, como administradores, contadores y empleados del operador logístico.
    c) Se asignan los permisos correspondientes a cada tipo de usuario.
    d) Los cambios se guardan y aplican en el sistema.

6.2. Fase 2: Implementación de los módulos

1. **Registrar transacciones financieras**: Los empleados registran transacciones financieras en Odoo.
    a) El empleado accede al módulo contable de Odoo.
    b) El empleado ingresa los detalles de la transacción, como ventas, compras, gastos e ingresos.
    c) Los datos se guardan en el sistema y se actualizan los registros contables.
2. **Gestionar inventario**: Los empleados gestionan el inventario de la tienda online en Odoo.
    a) El empleado accede al módulo de inventario de Odoo.
    b) El empleado verifica y actualiza las existencias de productos.
    c) Se registran las entradas y salidas de productos en el sistema.

6.3. Fase 3: Desarrollo de la API de comunicación

1. **Definir requisitos de la API**: Se definen los requisitos de información y los tipos de datos necesarios para la comunicación entre Odoo y el sistema del operador logístico.
    a) Los analistas de sistemas identifican los datos requeridos para la comunicación con el operador logístico, como información de pedidos, detalles de envío, seguimiento de paquetes, etc
    b) Se definen los formatos de datos, protocolos de comunicación y requisitos de autenticación necesarios para la API.
2. **Desarrollar endpoints de la API**: Los desarrolladores implementan los endpoints o puntos de acceso en la API para permitir la comunicación bidireccional entre Odoo y el sistema del operador logístico.
    a) Los desarrolladores crean y configuran los endpoints necesarios para enviar y recibir datos entre los sistemas
    b) Se establecen las rutas, métodos de solicitud (GET, POST, etc.), parámetros de entrada y salida, y se define la lógica de procesamiento de los datos
3. **Validar datos de la API**: El sistema realiza la validación de los datos intercambiados a través de la API para garantizar su coherencia y precisión.
    a) El sistema valida los datos recibidos a través de la API, verificando su estructura, formatos y consistencia
    b) Se aplican reglas de validación para asegurar que los datos cumplan con los requisitos establecidos antes de ser procesados y almacenados en Odoo.
4. **Realizar pruebas de integración**: Se llevan a cabo pruebas exhaustivas para verificar el correcto funcionamiento de la API y el intercambio adecuado de información entre Odoo y el sistema del operador logístico.
    a) Se diseñan y ejecutan casos de prueba para validar los diferentes escenarios de comunicación entre los sistemas.
    b) Se verifica que la API responda correctamente a las solicitudes, se intercambien los datos esperados y se manejen adecuadamente los errores.
    c) Se documentan los resultados de las pruebas y se realizan las correcciones necesarias antes de poner en producción la API de comunicación.

7 ORM DE ODOO

Este apartado más técnico está destinado a explicar, a bajo nivel, como funciona exactamente la comunicación entre el ORM de Odoo y la base de datos.
En la Figura 3 se muestra el diagrama de clases.
Explicación:

- **Odoo**: Representa la instancia principal de Odoo que proporciona métodos para interactuar con los modelos y la base de datos.
- **Model**: Define la estructura y el comportamiento de un modelo en Odoo. Los modelos representan entidades de negocio, como pedidos, productos, clientes, etc.
- **RecordSet**: Es una colección de registros pertenecientes a un modelo específico. Permite realizar operaciones de búsqueda, creación, escritura y eliminación en conjuntos de registros.
- **Record**: Representa un registro individual en un modelo. Proporciona métodos para leer, escribir y eliminar registros, así como acceder a sus campos y relaciones.
- **Environment**: Proporciona un entorno de ejecución para las operaciones del ORM de Odoo. Incluye la conexión a la base de datos a través del DBConnector y el acceso a los modelos y registros.

- **DBConnector**: Establece la conexión con la base de datos y permite ejecutar consultas y transacciones.

- **Connection**: Representa una conexión activa con la base de datos y se encarga de ejecutar consultas y realizar operaciones de actualización.

- **ResultSet**: Contiene los resultados de una consulta a la base de datos. Proporciona métodos para acceder a los datos y manipularlos.

El diagrama de flujo del ORM de Odoo muestra cómo interactúan estas clases para gestionar los modelos y la comunicación con la base de datos, facilitando las operaciones de búsqueda, creación, modificación y eliminación de registros.

8 Flujo de la integración

El diagrama de flujo de la API describe la comunicación entre el operador logístico y Odoo, permitiendo el intercambio de información y acciones relacionadas con los pedidos y productos. En la Figura 4 se muestra el diagrama de flujo. Explicación:

- **POST: Cliente del pedido:** El operador logístico realiza un envío a Odoo de todos los clientes que detecta que son nuevos en el sistema, para así los deja listos para posteriormente asignarlos a los pedidos de venta.

- **POST: Productos del pedido:** El operador logístico realiza un envío a Odoo de todos los productos que detecta que son nuevos en el sistema, para así los deja listos para posteriormente asignarlos a los pedidos de venta.

- **POST: Pedido nuevo:** El operador logístico realiza un envío a Odoo de cada orden que se ha generado en su sistema con tal de tener toda la información actualizada entre ambos sistemas.

- **POST: Validar pedido:** El operador logístico hace una llamada para validar que la orden es correcta. De esta manera, se valida el pedido de venta también en Odoo y se puede proceder con su preparación en el almacén.

- **GET: Pedido de venta:** El operador logístico envía una solicitud para conocer el estado del pedido de venta, en caso de haber sido validado correctamente en Odoo, puede proceder con el envío. En caso contrario, deberá o bien esperar a que se acabe de validar o ponerse en contacto con algún administrador de ventas del sistema para que pueda solucionar los problemas que pueda haber.

El diagrama de flujo de la API representa las principales interacciones entre el operador logístico y Odoo en el flujo de un pedido de venta. Este es el flujo más importante y, por lo tanto, debe ser el más robusto, puesto que si este falla, no se pueden registrar ni facturar todas las ventas que llegan.

9 Resultados

En este apartado se presentarán los resultados obtenidos a lo largo de todo el desarrollo del proyecto. El objetivo es reflejar todo el trabajo que se ha hecho para, así, después, sacar unas conclusiones.

9.1. Aplicación web de Odoo

Como principal resultado tenemos la aplicación web Odoo funcionando sin fallos y ya habiendo pasado por una configuración inicial de las principales aplicaciones. La curva de aprendizaje para poder lograrlo ha sido moderada, ya que, si bien al principio es necesario realizar cursos de formación básica, también es cierto que Odoo tiene la ventaja que es muy intuitivo y visual. De esta manera, se pueden ir haciendo pruebas, creando registros y viendo el comportamiento que tienen muy fácilmente.

Finalmente, se ha creado una infraestructura Docker con la imagen de Odoo dónde, a través de los archivos de configuración, se han definido servicios como el de la base de datos o el propio de Odoo, entre otros. Cabe mencionar que si, en un futuro, se quiere llevar este proyecto a un entorno de producción, se deberá hacer una configuración más precisa y más segura. Puesto que al haber trabajado enteramente en un entorno local, no se ha tenido en cuenta en ningún momento la seguridad e integridad del sistema.

9.2. Contabilidad

Tras finalizar formaciones de la aplicación de contabilidad y haber adquirido nociones básicas sobre contabilidad, se ha podido dejar configurado en el sistema todo lo esencial para empezar a trabajar en un entorno real. Cabe mencionar que la sincronización bancaria, punto que no entraba en los requisitos, no se ha realizado por razones obvias, a pesar de que el sistema está listo y preparado para hacerla.

9.3. Integración

Otro gran bloque del proyecto ha sido la integración de Odoo con un operador logístico. Este operador ha sido en todo momento ficticio, pero, al tener una documentación en condiciones con Swagger, es mucho más sencillo que cualquier cliente se adapte a las llamadas, obteniendo así una sincronización entre Odoo y un sistema externo.

Para la implementación, se ha desarrollado un módulo de Odoo desde cero basado en un módulo de la OCA llamado Base Rest. En este caso, la curva de aprendizaje ha sido más elevada, puesto que ha requerido alcanzar conocimientos nuevos acerca del ORM de Odoo, la arquitectura del módulo base y lenguajes de programación. Una ventaja y uno de los motivos por los que se decidió ejecutar el proyecto con Odoo, a nivel técnico, fue por las nociones que ya tenía de Python y Docker.

Así pues, como resultado de la fase de integración, se han obtenido los siguientes puntos:

- Módulo de Odoo funcional listo para ser instalado en cualquier base de datos

- Documentación de todas las llamadas disponibles en el sistema
- Documentación detallada acerca del desarrollo de este módulo, estandarizando procesos y dejando constancia de cómo se ha realizado
- Conocimiento acerca de la arquitectura y de la organización de ficheros de Odoo

9.4. Manuales

Un punto esencial a la hora de desarrollar un proyecto es generar documentación para el cliente, puesto que, al final, será quien utilice todos los días esta aplicación. Es por eso que, a medida que el desarrollo del proyecto iba concluyendo, la necesidad de tener unos manuales de usuario de los principales flujos y aplicaciones, iba a ser más que necesario.

Por eso, una de las últimas iteraciones de este proyecto ha sido generar esta documentación completa y visual para que, sin que el usuario tenga muchas nociones de Odoo, sea capaz de realizar las principales tareas. Han sido 4 manuales los que se han generado:

- **Flujo de compras:** El usuario final deberá ser capaz de completar este flujo de manera independiente, así que, en el documento, se detalla todo el recorrido que tiene una orden de compra, desde la creación de la misma en el sistema, hasta la creación y validación del recibo.
- **Flujo de ventas:** Este manual es muy similar al anterior, pero es igual de necesario, puesto que hay diferencias entre ambos flujos y es preciso tener los dos por separado.
- **Inventario:** En este documento se han intentado resaltar los puntos más importantes de la aplicación de inventario. Como punto más importante, los productos y la gestión de su stock. Explicando como se crean y actualizan los productos y como se procede cuando se quiere añadir disponibilidad de un producto manualmente.
- **Contabilidad:** A lo largo del documento se aconseja que el encargado de la aplicación de contabilidad sea un contable, puesto que hay muchos aspectos a nivel legal que se deben tomar en consideración. Es una introducción básica a la contabilidad de Odoo. Suficiente para realizar la gran mayoría de operaciones.

9.5. Fase de pruebas

Esta fase se ha ido manteniendo durante todo el transcurso del proyecto con tal de ir probando las nuevas funcionalidades configuradas y desarrolladas. A continuación, una lista de todos los procesos y funcionalidades que han sido sometidos a esta fase.

- Contenedor Docker de Odoo: El sistema ha sido sometido a una pequeña prueba de rendimiento para ver de lo que es capaz de soportar. Al no tener muchos procesos en paralelo y depender directamente de la potencia del sistema, los resultados han sido muy positivos. Aprovechando la API desarrollada, se ha hecho un volcado masivo de registros del modelo de Producto para ver como afectaba a la experiencia del usuario navegando por la interfaz y habiendo creado 10.000 registros, el sistema ha soportado correctamente. Se ha observado un ligero aumento de tiempo al hacer búsquedas en ese modelo, pero nada fuera de lo normal, teniendo en cuenta que la tabla en la base de datos tiene un mayor peso.
- Creación de apuntes contables: Con tal de validar que la contabilidad del sistema está bien configurada y es fiable, se han hecho distintas pruebas, creando facturas y recibos con diferentes impuestos, forzando comportamientos no permitidos por el sistema y validando que los datos teóricos cuadran exactamente con el sistema.
- Llamadas a la API: Para validar que realmente toda la parte de la integración está funcionando correctamente, se han realizado una serie de llamadas para cada modelo, forzando respuestas de error y respuestas válidas. Se ha hecho a través de Postman y ha tenido dos funciones:
  1. Validar que la respuesta es la deseada en todo momento: Como se han definido todas las llamadas, estén o no implementadas, es necesario validar que la respuesta que nos devuelve Odoo corresponde con la que teóricamente debería devolver.
  2. Comprobar los tiempos de respuesta: Esta ha sido más a nivel teórico, puesto que se han intentado seguir buenas prácticas de programación para que los tiempos de respuesta sean lo más bajos posible. Para realizar estas pruebas, se han recogido los tiempos de varias llamadas, distinguiendo entre modelo y tipo, y se ha hecho un cálculo medio de los tiempos de respuesta. En general, los resultados han sido buenos y óptimos. Son poco fiables, ya que se trata de un entorno local y una base de datos muy liviana.

Pese a no haber sido una fase de pruebas muy estricta, ha sido suficiente para poder validar que los procesos desarrollados cumplen con los requisitos y da a entender que todos los objetivos propuestos en el proyecto han sido logrados satisfactoriamente.

10 Conclusiones

En este último apartado se incluye un extracto de las conclusiones una vez finalizados todos los puntos del proyecto.

Para empezar, he podido verificar que la viabilidad de un negocio online es, a día de hoy, mucho más alta que un negocio físico. Las nuevas tecnologías ofrecen infinidad de soluciones y, por lo general, hay soluciones para cualquier cliente. Pese a que yo me he decantado por usar Odoo, es muy probable que existan herramientas y aplicaciones similares. Y con esto no quiero decir que los negocios físicos tienen los días contados ni mucho menos. Sin ir más lejos, este proyecto se ha planteado para un cliente que quiere empezar un negocio desde cero, pero es que, sin apenas hacer modificaciones, este se puede

adaptar a cualquier negocio físico que quiera expandirse y precisamente ahí quería llegar.

Aprender a desarrollar y utilizar herramientas que permitan gestionar negocios en línea es muy rentable, no solo a nivel económico, sino porque como está en constante evolución, nunca dejas de aprender. Todos los aprendizajes que he tenido a lo largo de proyecto me serán de utilidad, de eso no me cabe duda, pero, si tuviera que quedarme con un aprendizaje, el de las integraciones.

Hoy en día, al tener prácticamente todos los procesos en la nube, las integraciones entre sistemas está teniendo un auge muy grande en muchos sectores. Yo, personalmente, no había ni creado *endpoints* ni había hecho llamadas a una API. Me ha parecido muy interesante y curiosa la forma en la que puedes conectar dos sistemas que funcionan completamente diferente a través de estandarizar llamadas.

En definitiva, me he quedado con muy buen sabor de boca al finalizar el proyecto y puedo afirmar con total confianza que todo el tiempo dedicado e invertido ha sido recompensado con la satisfacción de tener todo perfectamente funcionando.

AGRADECIMIENTOS

Para concluir, me gustaría dedicar este apartado para agradecer a toda la gente que me ha ayudado en el proceso.

- A mi pareja, Diandra, por haberme soportado todo este tiempo, por haberme ayudado a sacar el lado bueno de las cosas y, sobre todo, por haberme apoyado en todo momento. Estoy seguro de que el proceso no hubiera sido el mismo sin ella.
- A mis padres, Ana y Santi, por haber confiado siempre en mí en todas mis decisiones y por haberme apoyado en cada una de ellas. Gracias a ellos soy quien soy.
- A mis gatos, Timmy y Gia, que han estado siempre a mi lado durante el desarrollo del proyecto.

REFERENCIAS

[1] http://en.wikibooks.org/wiki/LaTeX

[2] https://hub.docker.com/_/odoo

[3] https://odoo.com/es_ES/page/editions

[4] https://www.odoo.com/es_ES/app/accounting

[5] https://www.odoo.com/es_ES/app/sales

[6] https://www.odoo.com/es_ES/app/purchase

[7] https://hub.docker.com/_/postgres

[8] https://es.wikipedia.org/wiki/Odoo

[9] https://github.com/odoo/odoo

[10] https://es.wikipedia.org/wiki/GNU_Lesser_General_Public_License

[11] https://es.wikipedia.org/wiki/Docker_(software)

[12] https://trello.com

[13] https://mermaid.js.org/

[14] https://es.wikipedia.org/wiki/Requisito_funcional

[15] https://www.restapitutorial.com/httpstatuscodes.html

[16] https://www.odoo.com/documentation/15.0/es/applications/finance/accounting/get_started/chart_of_accounts.html

[17] https://www.odoo.com/documentation/15.0/es/applications/finance/accounting/taxes.html

[18] https://www.cybrosys.com/blog/orm-methods-in-odoo-15

[19] https://www.postman.com/

[20] https://github.com/OCA/rest-framework/tree/15.0/base_rest

[21] https://marshmallow.readthedocs.io/en/stable/

[22] https://github.com/Tecnativa/docker-whitelist

[23] https://www.canva.com/

[24] https://getquipu.com/blog/obligaciones-fiscales-tienda-online/

[25] https://www.infoautonomos.com/contabilidad/debe-haber-contabilidad-empresa/

[26] https://miro.com/

APÉNDICE

A.1. Diagrama de Gantt

Aquí se muestra el diagrama de Gantt correspondiente a la planificación del proyecto.

A.2. ORM de Odoo

Diagrama de clases de bajo nivel entre el ORM de Odoo y la base de datos.

A.3. Flujo de la API

Diagrama de flujo que muestra las llamadas disponibles en la API.

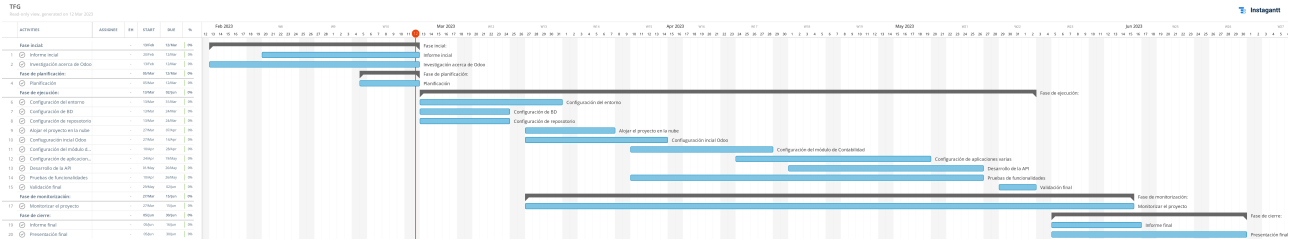

Fig. 2: Diagrama de Gantt

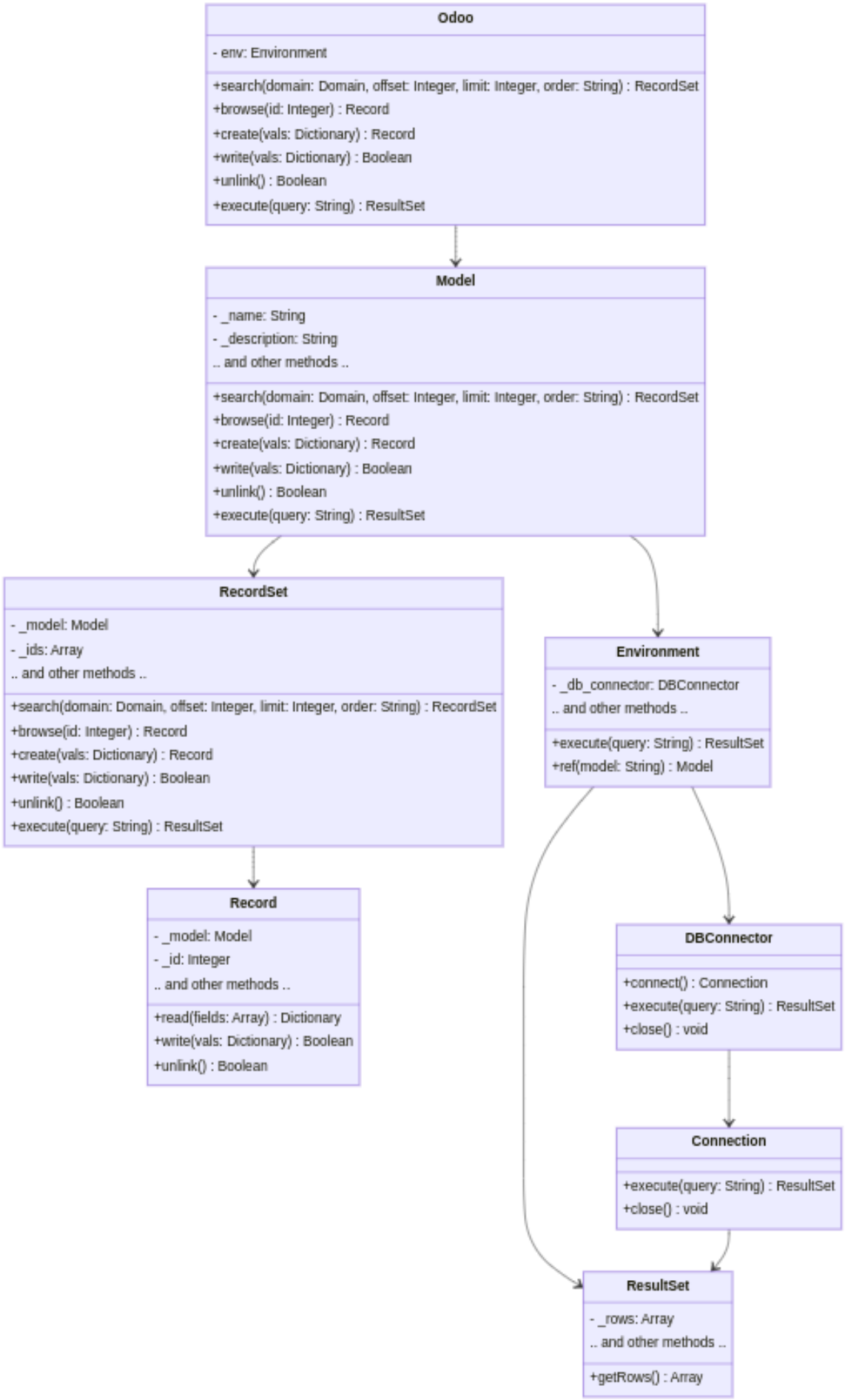

Fig. 3: ORM de Odoo

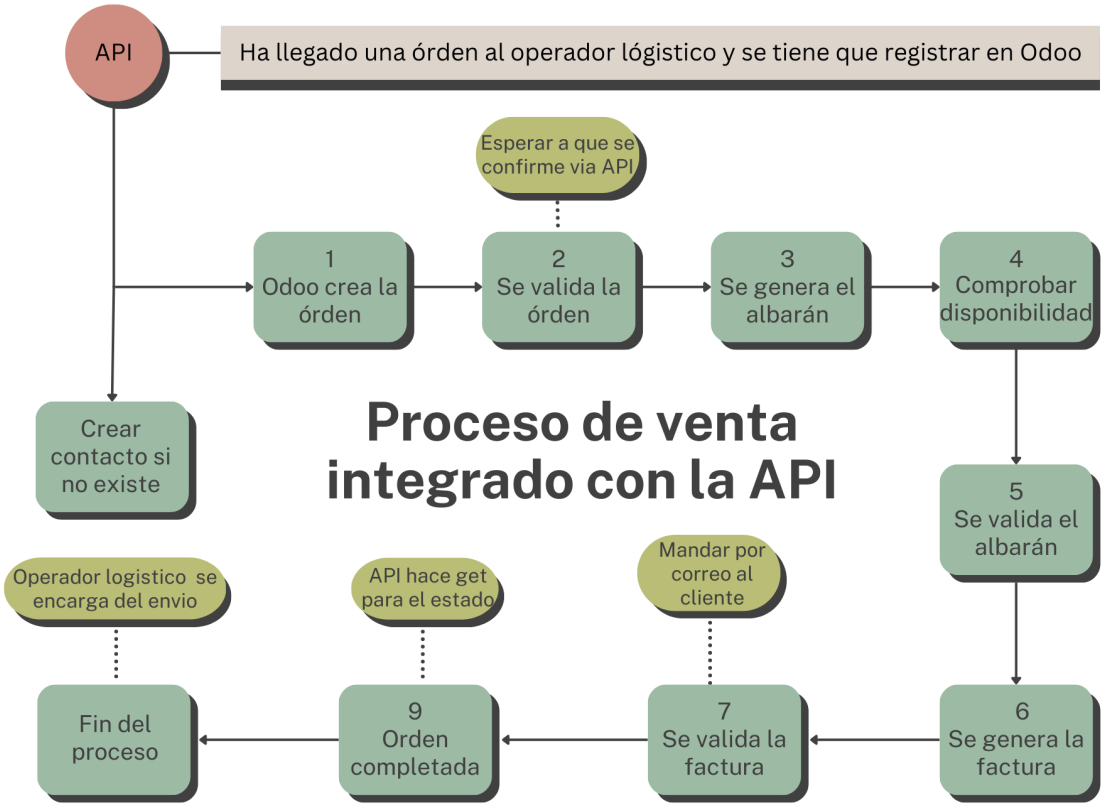

Fig. 4: Diagrama de flujo de la API