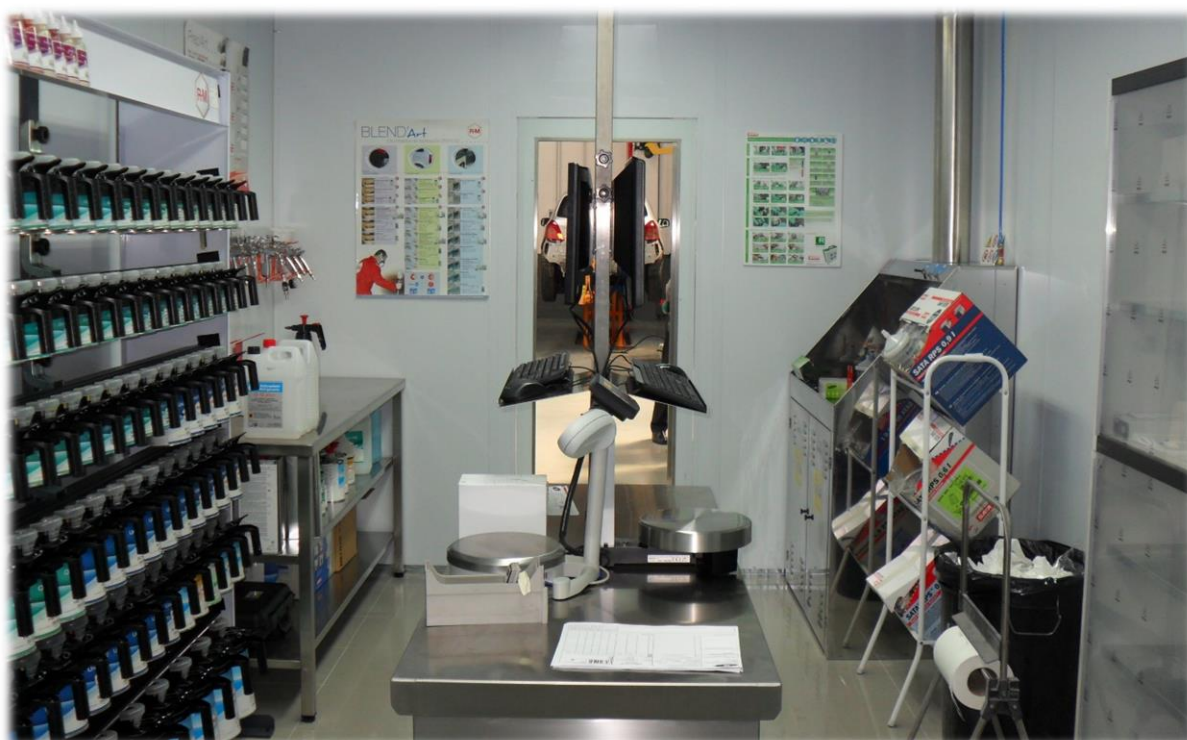


2º DAM 2024

# Easy Box

Tutor: Patricio Santiago Fernández Flórez

Proyecto Integrado fin de grado



Easy Box es un módulo creado dentro del ERP Odoo para la supervisión de las mezclas en la protección superficial aeronáutica industrial.

Enrique Raposo López

CESUR SEVILLA

## Tabla de contenido

1.	<a href="#">Introducción</a>	1
1.1.	<a href="#">Contextualización</a>	1
1.2.	<a href="#">Motivación</a>	1
1.3.	<a href="#">Objetivos</a>	2
1.4.	<a href="#">Estado del Arte</a>	2
1.5.	<a href="#">Glosario</a>	3
1.5.1.	<a href="#">Definiciones y acrónimos</a>	3
2.	<a href="#">Planificación</a>	2
2.1.	<a href="#">Metodología de desarrollo</a>	2
2.2.	<a href="#">Planificación del proyecto</a>	4
3.	<a href="#">Descripción del proyecto</a>	6
3.1.	<a href="#">Requisitos de alto nivel</a>	6
3.2.	<a href="#">Tecnologías usadas y Justificación de las Tecnologías</a>	6
3.3.	<a href="#">Usuarios finales</a>	7
3.4.	<a href="#">Evaluación de riesgos</a>	7
3.5.	<a href="#">Reducción de riesgos</a>	8
4.	<a href="#">Presupuestos</a>	8
4.1.	<a href="#">Costes directos</a>	8
4.2.	<a href="#">Costes indirectos</a>	9
4.3.	<a href="#">Otros costes</a>	9
4.4.	<a href="#">Coste total</a>	9
5.	<a href="#">Análisis de requisitos</a>	9
5.1.	<a href="#">Catálogo de actores</a>	9
5.2.	<a href="#">Requisitos funcionales</a>	10
5.3.	<a href="#">Requisitos no funcionales (ISO/IEC 25010)</a>	10
5.4.	<a href="#">Diagrama de los casos de uso</a>	11
6.	<a href="#">Diseño</a>	12
6.1.	<a href="#">Modelo conceptual de datos</a>	12
6.2.	<a href="#">Diagrama de interacción</a>	14
6.3.	<a href="#">Diseño lógico de la arquitectura</a>	15
6.4.	<a href="#">Diseño físico</a>	16
7.	<a href="#">Implementación del sistema</a>	17
8.	<a href="#">Pruebas del sistema</a>	21
8.1.	<a href="#">Pruebas unitarias</a>	21

8.2.	<a href="#">Pruebas de sistema</a>	23
9.	<a href="#">Conclusiones</a>	24
9.1.	<a href="#">Lecciones aprendidas</a>	24
9.2.	<a href="#">Objetivos cumplidos</a>	24
9.3.	<a href="#">Trabajo Futuro</a>	25
10.	<a href="#">Referencias bibliográficas</a>	26
11.	<a href="#">Anexos</a>	27
11.1.	<a href="#">Manual de usuario</a>	27
11.2.	<a href="#">Porciones de código explicado</a>	31

## Índice de figuras

1. <a href="#">Figura 1: ERP</a> .....	1
2. <a href="#">Figura 2: Kantan</a> .....	2
3. <a href="#">Figura 3: Diagrama de Gantt</a> .....	5
4. <a href="#">Figura 4: Tabla Costes Directos</a> .....	8
5. <a href="#">Figura 5: Tabla Costes Indirectos</a> .....	9
6. <a href="#">Figura 6: Tabla Otros Costes</a> .....	9
7. <a href="#">Figura 7: Tabla Coste Total</a> .....	9
8. <a href="#">Figura 8: Diagrama de casos de uso del usuario</a> .....	12
9. <a href="#">Figura 9: Diagrama de casos de uso del sistema</a> .....	12
10. <a href="#">Figura 10: Modelo conceptual de datos</a> .....	13
11. <a href="#">Figura 11: Diagrama de interacción del Administrador</a> .....	14
12. <a href="#">Figura 12: Diagrama de interacción del usuario</a> .....	15
13. <a href="#">Figura 13: Diseño lógico de la arquitectura</a> .....	16
14. <a href="#">Figura 14: Diseño físico de la arquitectura</a> .....	17
15. <a href="#">Figura 15: Estructura de directorios Easy Box</a> .....	18
16. <a href="#">Figura 16: Actualización de datos estándar de medición</a> .....	21
17. <a href="#">Figura 17: Prueba actualización de estándar error</a> .....	22
18. <a href="#">Figura 18: Introducción de datos de medición</a> .....	23
19. <a href="#">Figura 19: Sistema Warning</a> .....	24
20. <a href="#">Figura 20: Aplicaciones Fabricación</a> .....	27
21. <a href="#">Figura 21: Lista aplicaciones instaladas</a> .....	27
22. <a href="#">Figura 22: Menú administrador</a> .....	28
23. <a href="#">Figura 23: Detalles de un registro Normativa</a> .....	28
24. <a href="#">Figura 24: Elección nivel de operario</a> .....	29
25. <a href="#">Figura 25: Elección de normativa y muestra de referencia</a> .....	29
26. <a href="#">Figura 26: Inicio de sesión</a> .....	30
27. <a href="#">Figura 27: Gestión de usuarios/Cierre de sesión</a> .....	30
28. <a href="#">Figura 28: Información Easy Box</a> .....	31

## 1. Introducción

### 1.1. Contextualización

En la actualidad y desde el inicio de la fabricación de elementos aeronáuticos dentro de todo el país e incluso a nivel internacional, el mundo de la aeronáutica está compuesto por diferentes procesos secuencialmente desarrollados. Uno de ellos es el de la protección superficial o comúnmente conocido como “proceso de pintado” de las piezas individualmente, más concretamente este proceso se centra en las ciudades españolas de Sevilla e Illescas, aunque cada vez más empresas van incorporando este proceso en sus instalaciones con el objetivo de eliminar de la secuencia el transporte de las piezas a un lugar ajeno. Las empresas aeronáuticas suelen tener ya un sistema de gestión empresarial (Enterprise Resource Planning o ERP) para el control de su producción, pero muchas carecen de un módulo específico centrado en la tarea de hacer el mezclado de componentes de la pintura a aplicar y lo que conlleva.



Figura1: ERP // Fuente: [https://holocorp.com.mx/en\\_US/page/sistemas-de-gestion-empresarial-erp](https://holocorp.com.mx/en_US/page/sistemas-de-gestion-empresarial-erp)

### 1.2. Motivación

El motivo principal del desarrollo de este módulo proviene de la gran cantidad de tiempo que se consume en las plantas de producción aeronáuticas en la corrección de errores que se producen en una FR20 simplemente por el constante estrés generado por el gran volumen de trabajo y la falta de tiempo de los operarios en la consulta continua de la normativa de calidad vigente para el proceso a seguir.

Con EASY BOX será más fácil cumplir los requisitos de calidad vigentes del proceso en cuestión ya que en la actualidad no hay ningún software que avise de incompatibilidades. Hasta ahora era inspección de calidad quien hacia este trabajo de avisos y rehacer la inspección en caso de errores y una vez subsanados por producción.

### 1.3. Objetivos

El principal objetivo para la realización de esta aplicación es la de reducir tiempo y los consiguientes costes de producción en un sector que está constantemente en evolución normativa, siendo muy complejo el compaginar ser productivo y tener que leer e interpretar la normativa vigente en cada proceso que se realice además de:

- OBJ1. Crear un nuevo módulo dentro del ERP Odoo [1] con poder para crear perfiles a cada operario según categoría profesional y que Incluya datos relevantes de la normativa vigente en una base de datos que el operario debe conocer antes de la realización de la mezcla de componentes para la pintura
- OBJ2. Crear una aplicación que avise de incompatibilidades con la normativa
- OBJ3. Crear un nuevo módulo de Odoo [2] con una interfaz fácil e intuitiva para el operario

### 1.4. Estado del Arte

En la actualidad creía personalmente que no existía ninguna aplicación que realizara estas funciones descritas anteriormente, pero en la investigación he descubierto algunas aplicaciones que cumplen con requisitos de control de calidad, aunque específicamente no cumplen con esta idea tan concreta de mezclado de componentes en la pintura. Algunas de estas aplicaciones de calidad son:

- **Openinnova [3]:** Software de Gestión de Calidad ideal para llevar a cabo la implantación y mantenimiento de un Sistema de Gestión de Calidad (SGC) de cualquier tipo: ISO 9001, ISO 14001, OHSAS 18001... Logrando tener un control completo de la gestión de todo el sistema integrado.

- **Kantan [4]:** Al automatizar los procesos relacionados con las normas ISO 9001, ISO 14001, ISO 45001 e ISO 27001, el software permite a la empresa optimizar sus operaciones en términos de calidad, medio ambiente, salud y seguridad ocupacional, y seguridad de la información. Esta optimización conduce a una mayor

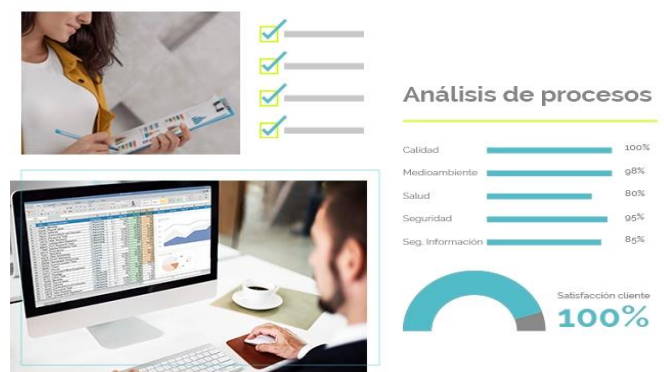


Figura 2: Kantan

eficiencia y productividad, ya que los empleados pueden centrarse en tareas más estratégicas en lugar de gestionar manualmente la conformidad con múltiples normas.

- **ShareMe QMS [5]:** Digitaliza la gestión documental y calidad. Automatiza procedimientos. Simplifica la gestión documental. Mejora el cumplimiento normativo

Existe en definitiva una multitud de software que gestionan un departamento de calidad en una empresa.

## 1.5. Glosario

### 1.5.1. Definiciones y Acrónimos:

- ERP: Enterprise Resource Planning. Siglas en inglés de un Sistema de Gestión Empresarial.
- FR20: Formato de Registro. Documento donde se registran manualmente todos los datos requeridos por la normativa en aplicación según proceso
- Normativa: Conjunto de pasos a seguir siguiendo unas especificaciones únicas dependiendo del proceso aeronáutico a desarrollar.
- MVC (Modelo Vista Controlador): MVC es una propuesta de arquitectura del software utilizada para separar el código por sus distintas responsabilidades, manteniendo distintas capas que se encargan de hacer una tarea muy concreta.
- Software: Conjunto de programas, instrucciones y reglas informáticas para ejecutar ciertas tareas en una computadora.
- Actor: Un actor es alguien o algo que interactúa con el sistema; es quien utiliza el sistema. Por la frase "interactúa con el sistema" se debe entender que el actor envía a o recibe del sistema unos mensajes o intercambia información con el sistema. Un actor puede ser una persona u otro sistema que se comunica con el sistema a modelar. En pocas palabras, el actor lleva a cabo los casos de uso. Un actor es un tipo (o sea, una clase), no es una instancia y representa a un rol. Gráficamente se representa con la figura de "stickman"
- PNC: Parte de No Conformidad. Documento oficial para la disputa entre instalaciones.

## 2. Planificación

### 2.1. Metodología de desarrollo

Para la realización de este proyecto he elegido la metodología INCREMENTAL ya que en un desarrollo iterativo e incremental el proyecto se planifica en diversos bloques temporales (de 15 días aprox. en este caso) llamados iteraciones. Las iteraciones se pueden entender como miniproyectos. En todas las iteraciones se repite un proceso de trabajo similar (de ahí el nombre "iterativo") para proporcionar un resultado completo sobre el producto final. En cada iteración se evoluciona el producto a partir de los resultados completados en las iteraciones anteriores, añadiendo nuevos objetivos/requisitos o mejorando los que ya fueron completados

Este modelo de desarrollo se utiliza cuando el cliente no sabe exactamente qué es lo que necesita y/o lo va sabiendo conforme va viendo cuales son los resultados del proyecto. El cliente necesita hacer cambios a corto plazo (nuevos requisitos o cambios en los ya realizados). No es



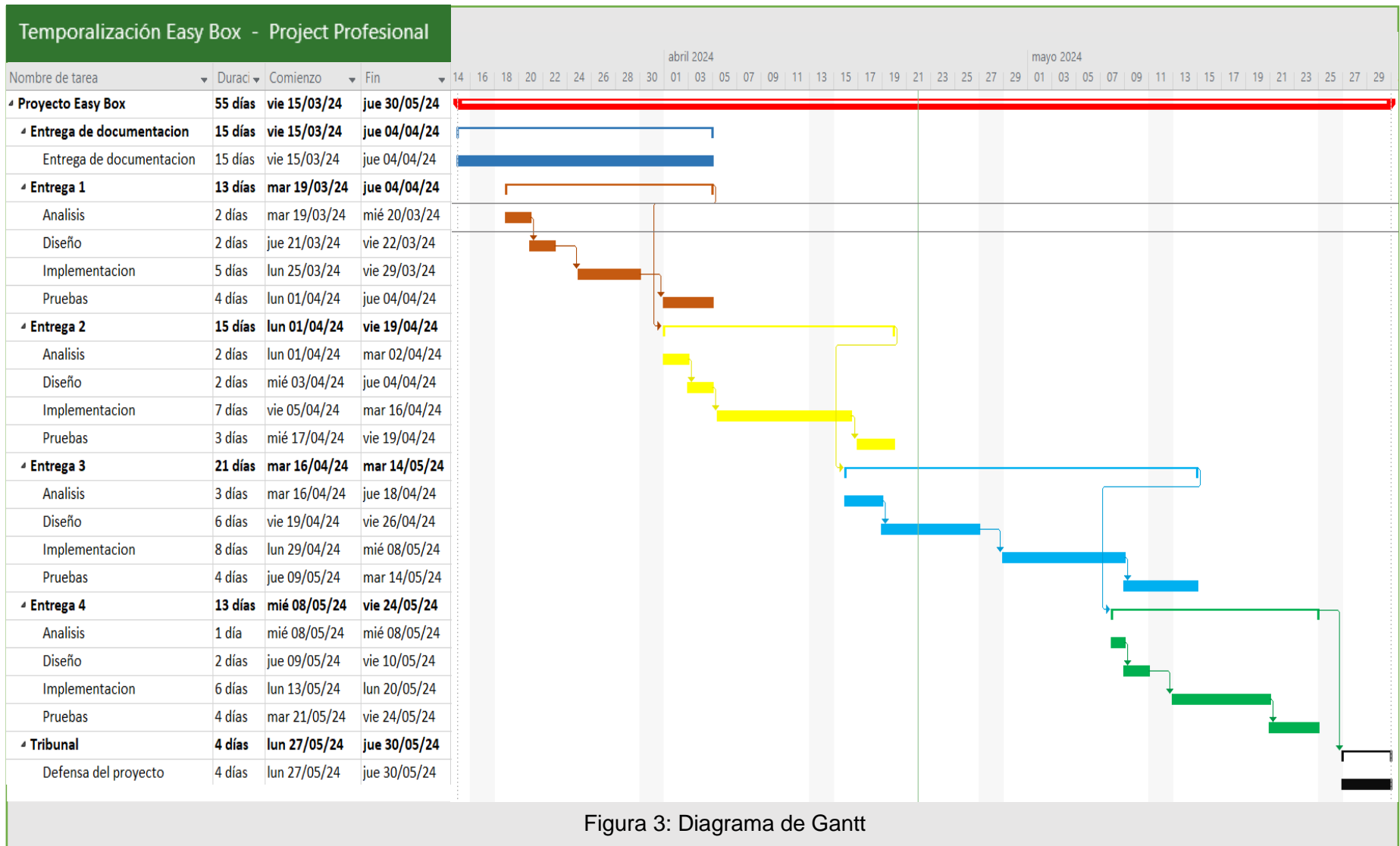
necesario realizar una recolección completa y detallada de todos los requisitos antes de empezar el desarrollo del proyecto. Con lo cual la secuencia a seguir en este proyecto es:

- Entrega de documentación: Este sprint inicial o previo consiste en hacer una entrega de documentos requeridos por el cliente/docente en el que se solicitan datos personales.
- Entrega1: Para el cumplimiento de la primera entrega se espera crear un nuevo módulo dentro del ERP de Odoo con el nombre Easy Box y la definición de los ficheros principales
- Entrega2: Esta entrega se centrará en crear ficheros para la creación de usuarios según categoría profesional e introducción de datos teóricos según normativa por el administrador.
- Entrega3: Esta entrega es para crear el sistema de detección de incompatibilidades y lanzamiento de avisos.
- Entrega4: Con esta entrega se crea una interfaz donde el operario puede iniciar sesión e introducir sus mediciones reales dependiendo de la normativa a seguir.
- Tribunal: Esta será la entrega final del proyecto al cliente y su instalación, a la vez de una explicación metódica de su funcionamiento.

## **2.2. Planificación del proyecto**

Para la realización de la temporalización he elegido realizarlo con Microsoft Project ya que es un software incorporado en el paquete office proporcionado por Cesur durante el tiempo del ciclo formativo. Con él he elaborado el siguiente diagrama de Gantt.





### 3. Descripción del proyecto

#### 3.1. Requisitos de alto nivel

Requisitos de alto nivel son las principales condiciones y/o características generales que debe cumplir el proyecto, así como los límites. Se trata de un listado en el que se estiman y priorizan los requisitos de alto nivel del Proyecto. Es **IMPORTANTE** hacer una buena distinción con los aspectos técnicos del desarrollo del proyecto que sería otra definición.

Incluyendo en su ERP el módulo “EASY BOX” las empresas dedicadas a la aplicación de protección superficial podrán tener una mejora en la producción reduciendo el tiempo de corrección de errores humanos producidos por el normal funcionamiento del proceso ya que son muchos los datos a tener en cuenta a la hora de hacer el proceso de pintado, desde el almacenamiento de materiales hasta el proceso de curado de la pieza pintada. Estos requisitos de alto nivel son definidos por los interesados en el Proyecto:

- RAN1.** Crear un software diseñado para ser usado a veces por operarios sin preparación técnica informática ninguna, con lo que ha de ser muy intuitivo.
- RAN2.** Crear una interfaz para que cada usuario deba ser registrado por el administrador incluyendo su categoría profesional para poder usar el software. El administrador será quien introduzca los datos teóricos del estándar según la normativa.
- RAN3.** Ha de ser un módulo integrado dentro del ERP Odoo y ser instalado en localizaciones estratégicas (Box de mezclas) de la nave industrial para aprovechar al máximo el tiempo empleado por el operario en su uso.
- RAN4.** Crear una interfaz donde se deba introducir datos de medición relativos al mezclado de componentes de la pintura y requeridos por el propio software para continuar si son válidos.
- RAN5.** En caso de introducirse un dato de medición incorrecto se comunicará al operario mediante un Warning o aviso de incompatibilidad para su corrección.

Los autores del proyecto debemos aprender a recopilar los requisitos del proyecto, con el suficiente grado de profesionalidad que asegure una exacta comprensión de las necesidades del proyecto, basada en las expectativas del cliente del Proyecto.

#### 3.2. Tecnologías usadas y Justificación de las Tecnologías

- Sistema de gestión empresarial Odoo en su versión Community: Se usará la versión Community de Odoo ya que es la gratuita del ERP y reduciría los costes de producción del proyecto.
- Visual Studio Code: Se usará para la edición del código en Python y XML en la creación del nuevo módulo tal como indica la documentación oficial de Odoo. Es un editor de

código fuente multiplataforma desarrollado por Microsoft. Incluye soporte para la depuración, control integrado de Git de forma nativa, resaltado de sintaxis, autocompleta el código de forma inteligente, etc. Es gratuito y de código abierto.

- ORM: Un componente clave de Odoo es el ORM. Esta capa evita escribir la mayoría del SQL manualmente y proporciona extensibilidad y servicios de seguridad
- Arquitectura MVC: MVC se usa inicialmente en sistemas donde se requiere el uso de interfaces de usuario. Su fundamento es la separación del código en tres capas diferentes, acotadas por su responsabilidad, en lo que se llaman Modelos, Vistas y Controladores.
- Lenguaje de programación Python: Odoo utiliza como lenguaje base Python, por lo que este será el lenguaje principal a usar tanto para la lógica de la aplicación como la de los modelos.
- Lenguaje de marcado extensible XML (XML por sus siglas en inglés) es un formato simple basado en texto para representar la información de manera estructurada. Se usará para definir las vistas del modelo MVC anteriormente descritas.

### 3.3. Usuarios finales

Los usuarios finales de este módulo de Odoo van a ser los operarios dedicados a la preparación de las mezclas de una nave aeronáutica especializada en aplicación de pinturas de protección superficial. En un principio no se puede expandir a otros sectores (Automoción, Construcción, etc.) ya que carecen de estándares que regulen este proceso.

### 3.4. Evaluación de riesgos

Con una aplicación tan específica los riesgos para su incorporación en el normal funcionamiento de la empresa pueden ser:

- ER1. Un riesgo es el límite en la entrega del proyecto, es decir, no ser capaces de entregar el proyecto en la fecha indicada.
- ER2. Rechazo por parte de los operarios, ya que al ser una nueva tecnología incluye una disminución temporal de tiempo productivo.
- ER3. La necesidad de tener la base de datos constantemente actualizada a la normativa que evolucione, sin dilación en el tiempo, ya que dejaría de ser eficiente el módulo.
- ER4. El deterioro del material táctil incluido en el box de mezclas por el normal uso.
- ER5. Pérdida de la conexión con la base de datos y por consiguiente errores en el normal funcionamiento de la aplicación.

### 3.5.Reducción de riesgos

Para una reducción de los riesgos a su mínima expresión podemos emprender las siguientes acciones:

- RR1. Para mitigar el [\[ER1\]](#) se debe hacer una buena planificación y ser eficientes en el trabajo estipulado con el diagrama de Gantt.
- RR2. Realizar cursos de formación con los operarios que puedan utilizar la aplicación incluso a nivel practico. Con esta acción se podrá mitigar [\[ER2\]](#)
- RR3. Dedicar un tiempo a la actualización de la base de datos por parte del director/a de calidad que es a quien notifican de las evoluciones en los estándares normativos y a qué datos afecta. Con esto se puede mitigar el [\[ER3\]](#)
- RR4. Para la mitigación de [\[ER4\]](#) hay que cambiar el material deteriorado si fuera necesario
- RR5. Tener una buena conexión y mantenimiento de la red local sería una buena opción para mitigar al máximo el [\[ER5\]](#)

## 4. Presupuestos

### 4.1.Costes directos

Los costes directos para el desarrollo de un módulo completo de Odoo con los objetivos propuestos incluidas todas y cada una de las fases del desarrollo, serán los del tiempo usado en su implementación que están muy relacionados con el diagrama de Gantt anteriormente expuesto.

Se estima que en la “entrega 1” se consumirán 20 horas en la que se construirá la estructura inicial del módulo y la configuración de los archivos principales para que ya aparezca como nuevo módulo dentro del ERP incluido en la categoría o sección de “Fabricación”. La “entrega 2” en la que se implementará el sistema de inicio de sesión y la creación de las entidades que van a interactuar con la aplicación, durará 25 horas. En la “entrega 3” se usará 30 horas y se centrará en la creación del sistema de aviso de incompatibilidades y para la “entrega 4” se necesitará 30 horas que se centrarán en el desarrollo de las vistas de los usuarios del sistema y presentación del proyecto al cliente/tribunal. Como será un solo desarrollador quien lo realice en solitario durante un tiempo estimado total de 105 horas a un precio de 18€/hora, el coste directo es de 1890€ con el IVA incluido.

Fase del desarrollo	Tiempo	Coste
Documentación + Entrega 1	20 horas	360€
Entrega 2	25 horas	450€
Entrega 3	30 horas	540€
Entrega 4 + Tribunal	30 horas	540€

Figura 4: Tabla Costes Directos.

## 4.2. Costes indirectos

Los costes indirectos serán los derivados del normal funcionamiento tanto del equipo informático (Hardware), los softwares usados, como los ocasionados en el entorno de trabajo.

Elemento	Coste
Ordenador portátil	88€
Windows 11	2€
Visual Studio Code	0€
Odoo	0€
Microsoft Office	12€

Figura 5: Tabla Costes Indirectos.

## 4.3. Otros costes

En este apartado se han de incluir los costes ocasionados por el normal funcionamiento del equipo de desarrollo.

Servicio	Coste
Luz	85€
Agua	12€
Conexión a internet	32€

Figura 6: Tabla Otros Costes

## 4.4. Coste total

Haciendo una suma de los costes directos, indirectos y otros costes tenemos un total de:

Concepto	Coste
Coste directo total	1890€
Coste indirecto total	102€
Otros Costes total	129€
<b>Total de costes</b>	<b>2121€</b>

Figura 7: Tabla Coste Total

# 5. Análisis de requisitos

## 5.1. Catálogo de actores

- Rol de Usuario: cuando se inicie sesión como administrador, será la persona encargada de poder crear los nuevos usuarios con su categoría profesional y a su vez la de introducir los datos relevantes de la normativa y sus futuras evoluciones si fuera necesario. A nivel profesional será la persona responsable del departamento de calidad. En cuanto al Usuario

como operario, será realmente la persona que usará la aplicación la mayor parte del tiempo. Será el operario de la nave industrial encargado de hacer la preparación y mezcla de la pintura según proceso a seguir.

- Rol sistema “Easy Box”: Será el responsable de lanzar los avisos de incompatibilidades con el estándar de medición a través de una ventana modal.

## 5.2. Requisitos funcionales

RF1. Inicio de sesión (Satisface [RAN2](#))

- Será necesario que cada usuario introduzca su nombre de usuario, número de operario y contraseña para iniciar sesión.

RF2. Actualización de datos estándares (Satisface [RAN2](#))

- Será el administrador quien tenga capacidad de introducir y/o modificar datos teóricos relativos a la normativa vigente.

RF3. Creación de nuevos usuarios habilitados para usar la app (Satisface [RAN2](#))

- La aplicación dará capacidad de crear usuarios al administrador

RF4. Introducción de datos de medición (Satisface [RAN4](#))

- La aplicación solicitará temperatura y humedad además de otras mediciones en el box de mezclas en el momento de hacer la preparación de la pintura
- Solicitará tiempo de reacción de la mezcla
- Solicitará proporción de los componentes
- La aplicación dará opción de elegir al operario la normativa a referenciar

RF5. Aviso de incompatibilidades (Satisface [RAN5](#))

- La app lanzará un aviso en el momento que se intente validar un dato que no esté dentro de unos valores establecidos en la normativa

## 5.3. Requisitos no funcionales (ISO/IEC 25010)

Los requisitos no funcionales relacionados con el software corresponden a los que dicta la norma ISO 25010 [\[6\]](#) que los define como un conjunto de requisitos y criterios de evaluación que permiten medir la calidad de un producto de software y se puede desglosar en los siguientes apartados:

- **Fiabilidad:** Capacidad de mantener su nivel de rendimiento en condiciones normales de uso o capacidad del producto software para recuperar los datos directamente afectados y reestablecer el estado deseado del sistema en caso de interrupción o fallo.

- **Usabilidad:** Se define como la capacidad de un software de ser comprendido, aprendido, usado y ser atractivo para el usuario. Este concepto es el Core de Easy Box para con sus usuarios, a veces sin una formación técnica adecuada con lo que para cumplir uno de los requisitos más importantes para el cliente final que es el de ahorrar costes de producción reduciendo el tiempo empleado por los operarios en el registro de mediciones que posteriormente serán precisados por el departamento de garantía de calidad, nuestra app cumpliría con el [RAN1](#) al ser intuitiva para que el operario no tuviera que pararse mucho tiempo en escribir manualmente los datos medidos en una FR20. Igualmente, la app quedará instalada en un hardware táctil facilitando su usabilidad en el box de mezclas y así cumpliendo el [RAN3](#).
- **Eficiencia:** Capacidad de utilizar los recursos de forma óptima para lograr los resultados deseados, es decir, la eficiencia es el grado en el que Easy Box emplea óptimamente los recursos del sistema Odoo para lograr su objetivo.
- **Mantenibilidad:** Capacidad de ser modificado, corregido o mejorado de manera eficiente.
- **Portabilidad:** Se define como la característica que posee un software para ejecutarse en diferentes plataformas, es decir, la portabilidad se refiere exclusivamente a la propiedad que posee Easy Box que le permite ser ejecutado en diferentes plataformas y/o sistemas operativos como Linux y Mac OS entre otros.
- **Seguridad:** Capacidad de proteger la información y los recursos del sistema.
- **Compatibilidad:** Capacidad de interactuar con otros sistemas o componentes sin problemas, se identifica como la condición que hace que dos elementos puedan comprenderse.

#### 5.4. Diagrama de los casos de uso

En el diagrama de casos de uso, las funciones del sistema en cuestión se representan desde el punto de vista del usuario (llamado “actor” en UML). Este actor no tiene que ser necesariamente un usuario humano, sino que el rol también puede atribuirse a un sistema externo que accede a otro sistema. De este modo, el diagrama de casos de uso muestra la relación entre un actor y sus requisitos o expectativas del sistema. En la práctica, esta estructura es adecuada para representar claramente las funciones y/o objetivos más importantes de un sistema. [\[7\]](#)



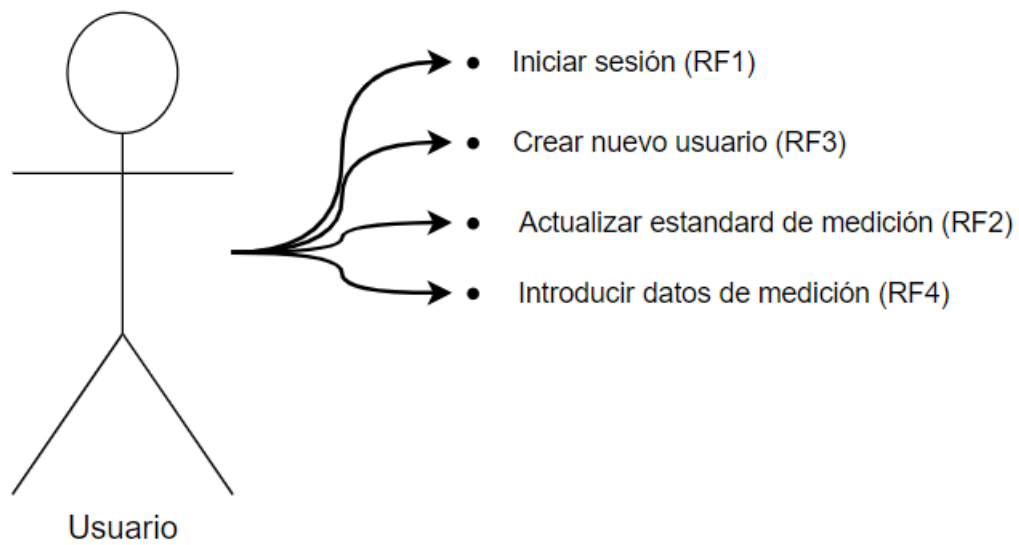


Figura 8: Diagrama de casos de uso del Usuario

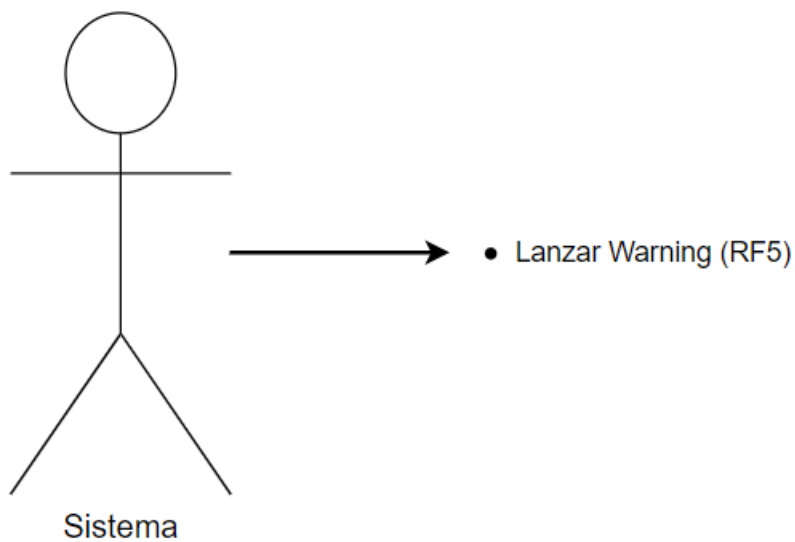


Figura 9: Diagrama de casos de uso del Sistema

## 6. Diseño

### 6.1. Modelo conceptual de datos

Los modelos conceptuales son a menudo abstracciones de cosas en el mundo real, ya sean físicas o sociales. El modelo conceptual de datos es un esquema que representa la estructura del sistema de información desde el punto de vista de los datos, es decir, las dependencias o relaciones

entre los diferentes datos del sistema de información: el modelo conceptual de datos (MCD) tiene por objeto el describir formalmente los datos que usará el sistema de información. Los conceptos básicos son: entidad (u objeto o clase), relación o asociación, propiedades o atributos, e identificador.

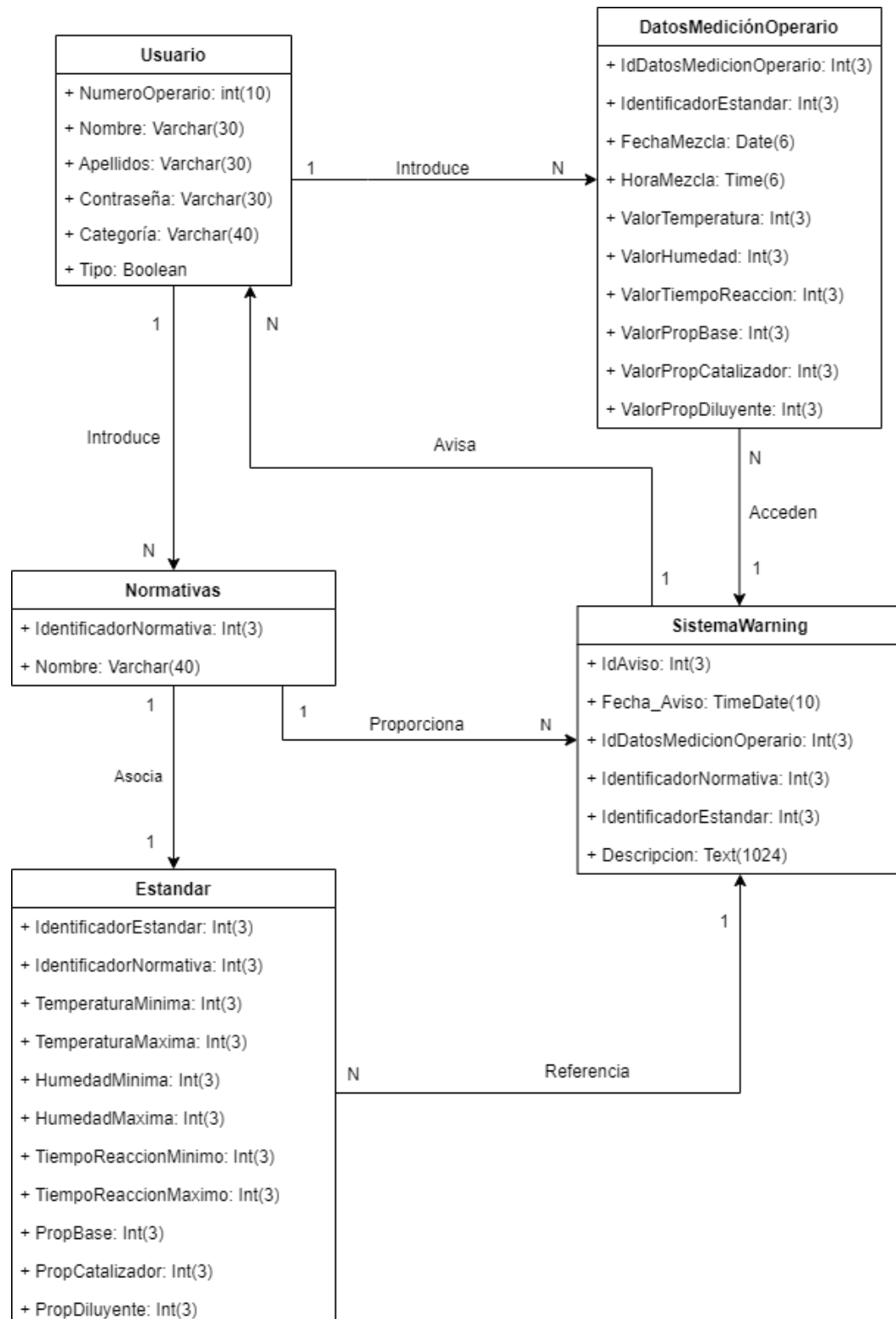


Figura 10: Modelo conceptual de datos

## 6.2. Diagrama de interacción

Un diagrama de interacción describe en detalle un determinado escenario de un caso de uso (llamados en este proyecto “Requisito Funcional”). En él se muestra la interacción entre el conjunto de objetos que cooperan en la realización de dicho escenario o dicho de otra forma los diagramas de interacción son representaciones *visuales* que muestran cómo diferentes elementos interactúan entre sí en un sistema o proceso. Estos diagramas son comúnmente utilizados en el campo de la ingeniería de software para modelar y entender la dinámica de sistemas complejos. En los siguientes ejemplos se harán referencias a los [RF1](#) Y [RF5](#) en los que se cumplen los requisitos de aviso de incompatibilidades con el estándar establecido y el inicio de sesión por parte de los usuarios.

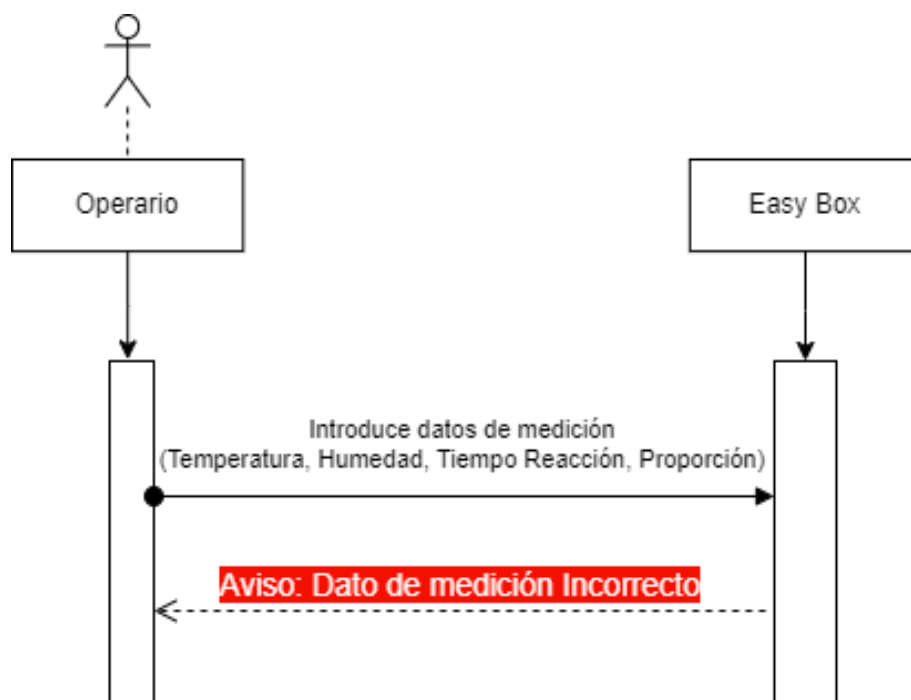
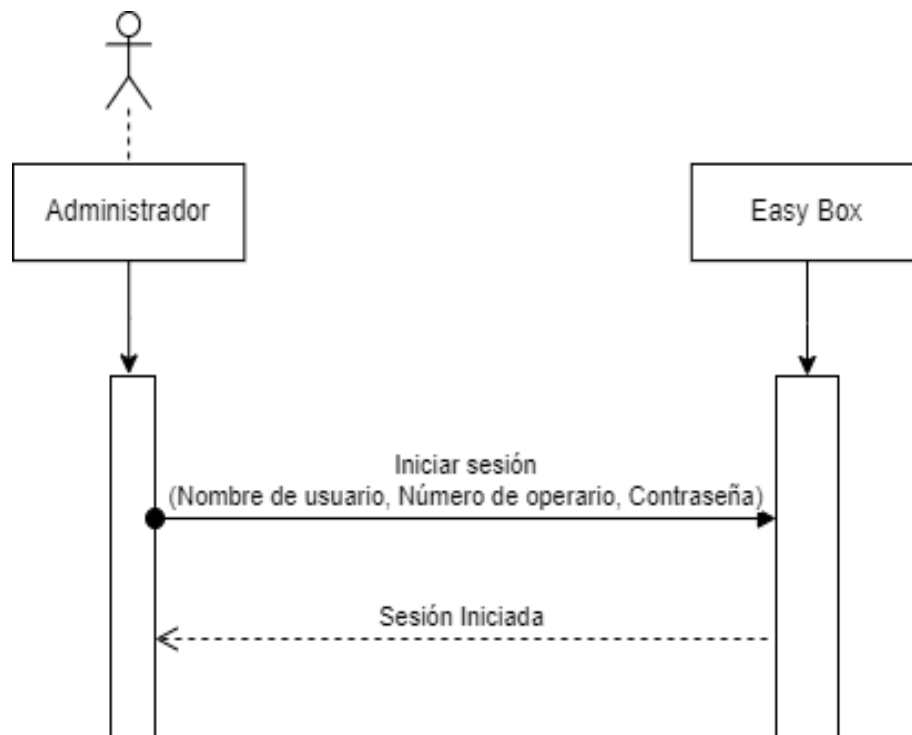


Figura 11: Diagrama de interacción [RF5](#)

Figura 12: Diagrama de interacción [RF1](#)

### 6.3. Diseño lógico de la arquitectura

El diseño lógico de un sistema de software se refiere a la etapa en el desarrollo de software donde se establece la estructura y la lógica del sistema sin entrar en detalles de implementación específicos. En esta fase, se define la arquitectura general del sistema, incluyendo la forma en que los distintos componentes interactúan entre sí para lograr los objetivos del software.

El diseño lógico del software proporciona una descripción conceptual del sistema, que sirve como guía para la implementación posterior del sistema en un lenguaje de programación específico, asegurando su estructura, mantenibilidad y cumplimiento de los requisitos del cliente.

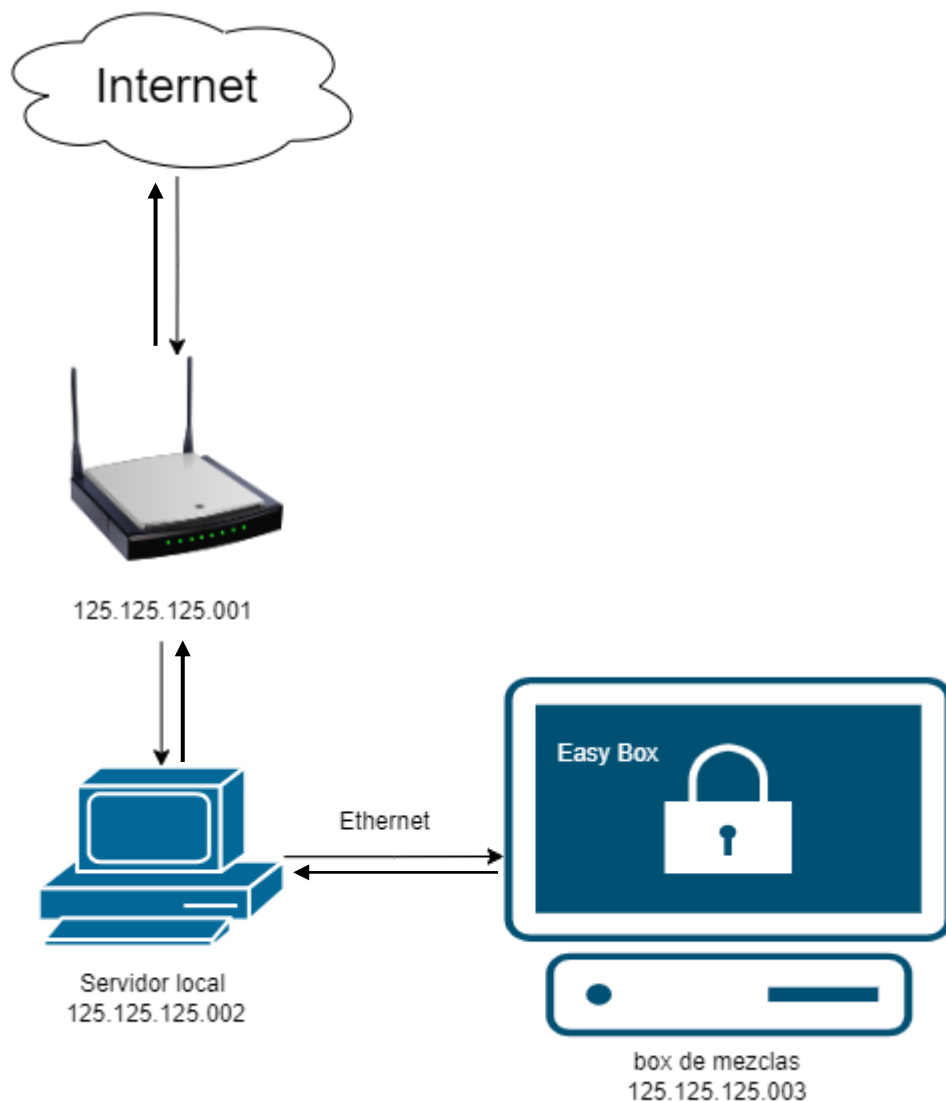


Figura 13: Diseño lógico de la arquitectura

#### 6.4. Diseño físico

El diseño físico de sistemas es la forma en que se lograrán las tareas del sistema, lo que incluye la manera de conjuntar sus componentes y las funciones que realizará cada uno de éstos. En el diseño físico se especifican las características de los componentes del sistema requeridos para poner en práctica el diseño lógico.

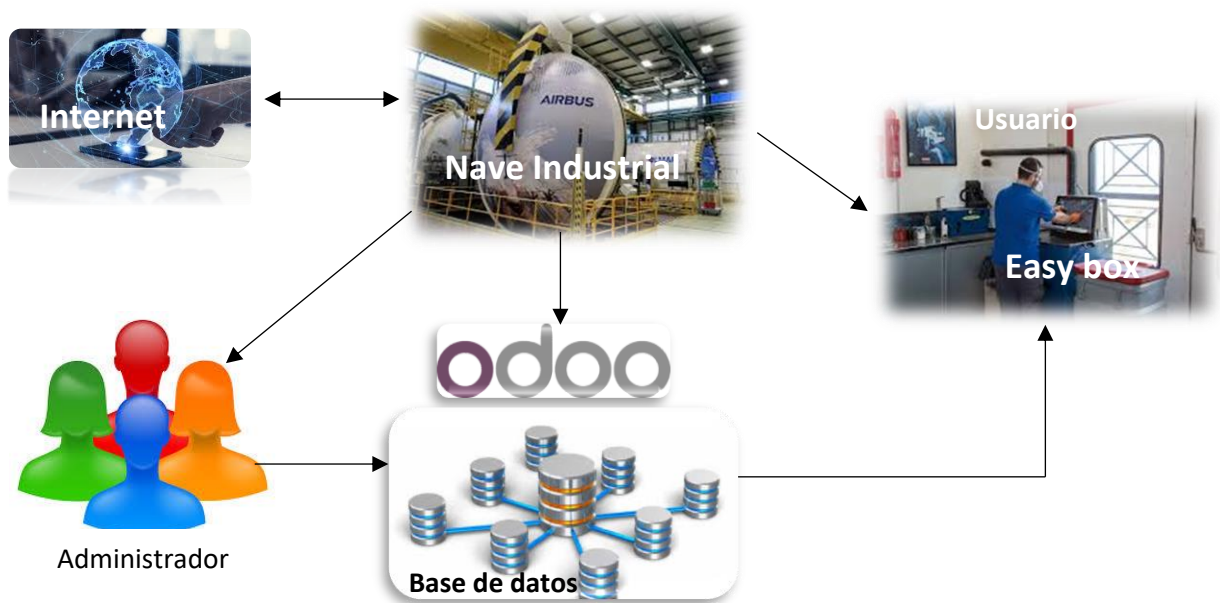


Figura 14: Diseño físico de la arquitectura

## 7. Implementación del sistema

En este apartado se va a hacer una explicación de cómo se ha desarrollado la creación del sistema de aviso de incompatibilidades integrado en el ERP Odoo al que hemos llamado “Easy Box”.

### a. Instalación ERP Odoo.

Es importante mencionar que usaremos principalmente el lenguaje de programación Python ya que Odoo se basa en él y para los archivos de definición de las vistas se usará el lenguaje de marcado XML.

Como primer paso es la instalación del propio sistema de gestión empresarial Odoo donde incorporaremos nuestro módulo dentro del directorio de carpetas que se nos crea en la instalación de Odoo y tenemos que seguir unos pasos dictados en su documentación oficial [\[1\]](#) (disponible en su página web) para la creación de nuestra estructura del módulo personalizado.

### b. Creación de la estructura principal.

Una vez instalado comenzamos usando el ejecutable “odoo-bin” seguido del comando scaffold para crear nuestra estructura principal del módulo dándole el nombre que queramos que sea el de nuestra nueva carpeta con la estructura. Cuando tenemos creado nuestra estructura ya podemos comenzar a editar el código para añadirle características y funcionalidades, para ello vamos a usar

el editor de código “Visual Studio Code”. Véase la siguiente figura donde se muestra la estructura de directorios creada por defecto.

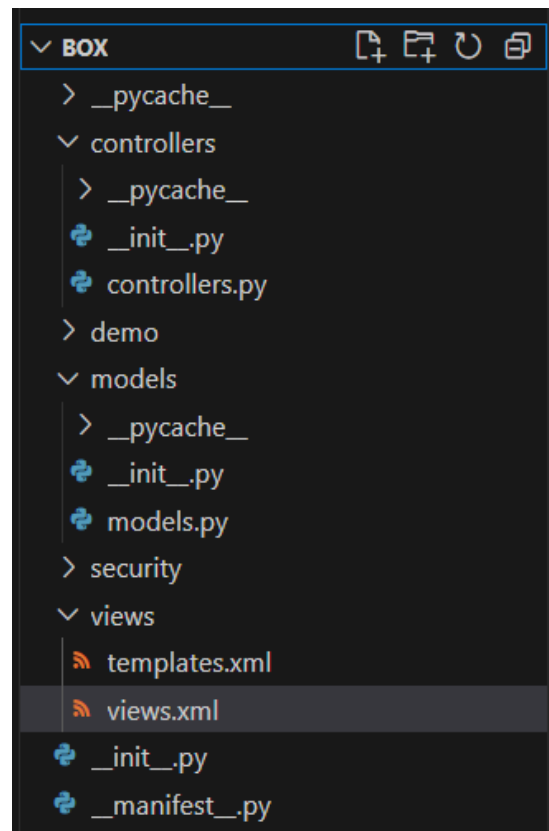


Figura 15: Estructura de directorios Easy Box

En esta estructura de directorios nos encontramos con varias carpetas principales que es donde implementaremos las funcionalidades descritas en los [RAN](#).

### c. Archivos principales

Siguiendo el modelo MVC entendemos como archivos principales a los ficheros donde se tendrá que desarrollar código para completar la funcionalidad de nuestro sistema y en concreto son:

#### **Models.py:**

Contenido en la carpeta models, es donde implementaremos las clases que representan tablas de nuestra base de datos donde se almacenará los datos y las funciones o métodos para crear las acciones de nuestra aplicación. Es de vital importancia que al comienzo del fichero se importen las librerías que se van a referenciar desde las clases “BoxNormativa” y “BoxUsuario” creadas para nuestra aplicación mediante el comando “`From odoo import models, fields, api, exceptions`”. Cada clase lleva asociado unas líneas fundamentales sin las cuales el sistema no se podría usar que son por ejemplo de la clase BoxNormativa:



`_name = 'box.normativa'` Esta línea es con la cual deberemos referenciar nuestra clase la cual es distinta de la siguiente: `name = fields.Char(string='Nombre de la Normativa', required=True)` que será el nombre principal que reciba la tabla donde guardaremos los datos con el nombre que le queramos dar a cada campo mediante su nombre y tipo de dato a almacenar además de los atributos que queramos añadir. Una vez creado cada campo podemos crear las funciones como, por ejemplo:

```
#función para actualizar la fecha y hora en la que se modifica la normativa
(cualquier registro)
def write(self, values):
    if 'fecha_hora_mezcla' not in values:
        values['fecha_hora_mezcla'] = fields.Datetime.now()
    return super(BoxNormativa, self).write(values)
```

Esta función concretamente almacena la fecha y hora en la cual se creó o se modifica cualquier registro.

### Views.xml:

Contenido en la carpeta views, es en este archivo donde se implementará el código que representa de forma visual lo almacenado en el fichero models.py o lo que es lo mismo, se implementa lo que verá el usuario de la aplicación. En esta parte del desarrollo se deben incluir sentencias como:

```
<!--datos a incluir por el operario-->
<sheet>
  <div class="oe_title">
    <h1>Mediciones del Operario</h1>
  </div>
  <group>
    <field name="fecha_hora_mezcla" readonly="1"/>
    <field name="name"/>
    <field name="operario_number"/>
    <field name="categoria_profesional"/>
    <field name="nivel_profesional"/>
    <field name="normativa_id"/>
  </group>
  <group>
    <field name="temp"/>
    <field name="humedad"/>
    <field name="tiempo_reaccion"/>
    <field name="proporcion_base"/>
    <field name="proporcion_catalizador"/>
    <field name="proporcion_diluyente"/>
  </group>
</sheet>
<!--Hasta aquí es para introducir mediciones-->
```

Estas líneas de código serán las responsables de mostrar un formulario por pantalla donde el operario debe introducir sus mediciones. Con la etiqueta “Field” se crea la vista de cada campo del modelo al que llamamos mediante el atributo “name”. Las etiquetas de “group” y “sheet” se usan para estilizar la vista y hacerla más intuitiva.

### Controllers.py:

Contenido dentro de la carpeta controllers encontramos el fichero controllers.py que es donde desarrollaremos la lógica de nuestra aplicación o dicho de forma coloquial es el encargado de enlazar las vistas con los modelos haciendo que tenga sentido lo expuesto visualmente con lo almacenado. En esta parte de la aplicación (también en lenguaje Python) se añade la clase BoxController donde se definen los métodos que entrarán en acción cuando el usuario presione en las vistas ciertos menús que requieren que se muestre por pantalla valores almacenados en la base de datos tanto de las normativas como datos incluidos por operarios.

```
from odoo import http
from odoo.http import request

class BoxController(http.Controller):

    @http.route('/box/admin', type='http', auth='user', website=True)
    def admin_page(self, **kwargs):
        return request.render('box.box_admin_form')

    @http.route('/box/operario', type='http', auth='user', website=True)
    def operario_page(self, **kwargs):
        #return request.render('box.box_user_form')
        normativas = request.env['box.normativa'].search([])
        return request.render('box.box_user_form', {'normativas': normativas})
```

Otros de los **ficheros externos** a estas tres carpetas principales encontramos el `__init__.py` y `__manifest__.py` fundamentales para que el propio sistema Odoo pueda reconocer nuestro módulo e integrarlo en su funcionamiento además del directorio “security” donde podremos crear los permisos necesarios a cada usuario del sistema.

Entrando en profundidad y en concreto el archivo `__manifest__.py` es quién proporcionará el nombre al módulo, el autor del mismo, una descripción de su operatividad, otros módulos de quien depende, integrar el nuestro en una categoría predefinida y otras características.

Una vez descrita la estructura principal de carpetas haremos una descripción de los archivos contenido en ellas que son de más relevancia en el apartado [anexos](#) cuando se haga la explicación de porciones de código.

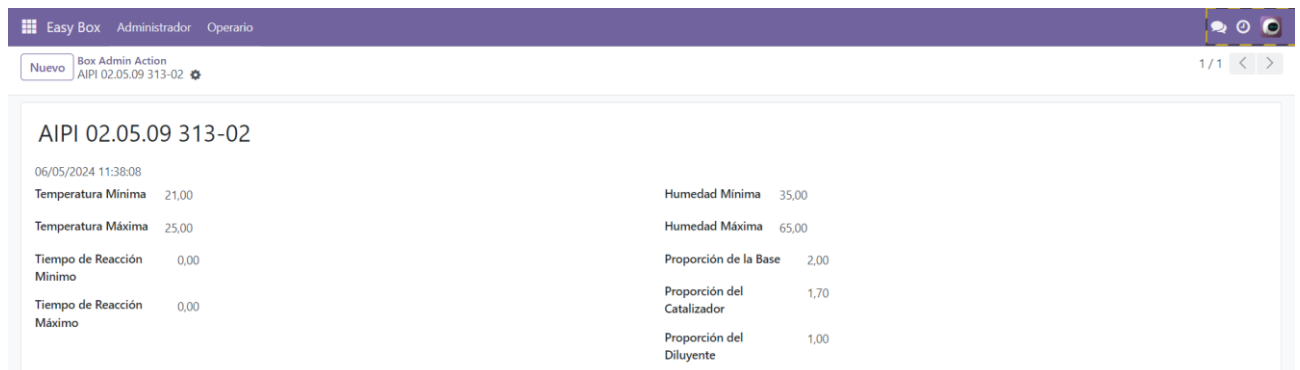
## 8. Pruebas del Sistema

En este apartado vamos a centrarnos en las pruebas que se han realizado en la implementación del sistema para comprobar que funciona correctamente.

### 8.1. Pruebas unitarias

#### PU1. Actualización de datos estándares de medición:

- *Descripción:* En esta prueba se comprobará que se satisface el requisito funcional 2 ([RF2](#)). Probaremos siendo administrador a introducir nuevos datos estándares o también llamado normativa.
- *Precondiciones:* El usuario que ha iniciado sesión debe ser el administrador (quien es el único habilitado para poder actualizar e incluir nuevos datos de estándares de medición) del sistema. Pulsando en el botón NUEVO se lanzará la acción de incluir nuevo registro, o entrando en un registro ya existente podrá modificarlo.
- *Resultado esperado:* Tras incluir nuevos datos estándar, los datos se deberán guardar sin presentar ningún error.
- *Resultado de la prueba:* Prueba pasada con éxito.

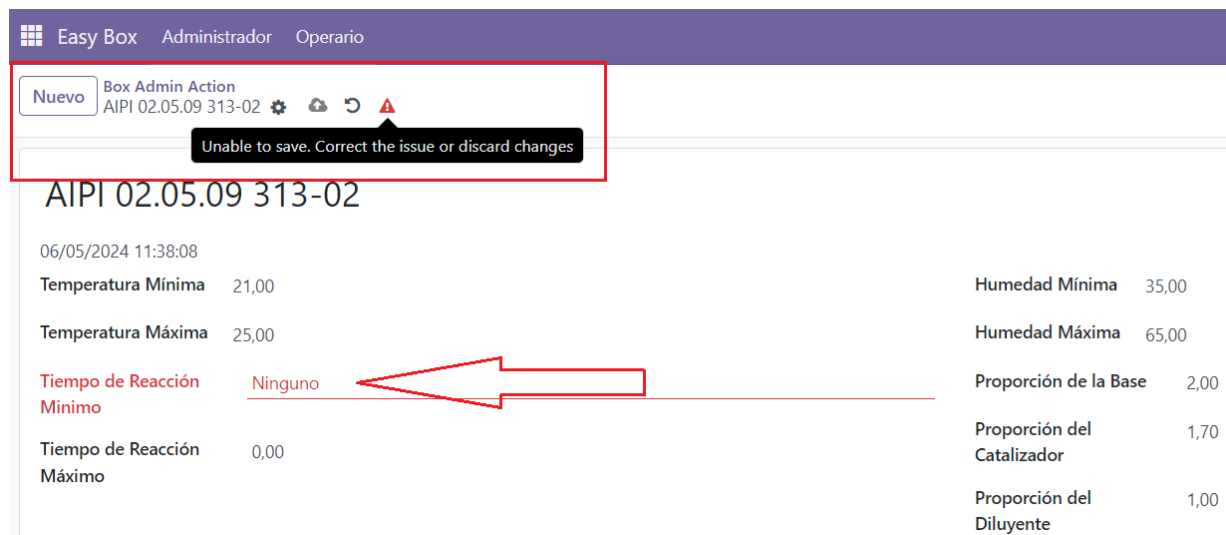


The screenshot shows the 'Easy Box' system interface. At the top, there is a navigation bar with 'Easy Box', 'Administrador', and 'Operario'. Below this, a 'NUEVO' button is visible. The main content area displays a table with the following data:

API 02.05.09 313-02	
06/05/2024 11:38:08	
Temperatura Mínima	21,00
Temperatura Máxima	25,00
Tiempo de Reacción Mínimo	0,00
Tiempo de Reacción Máximo	0,00
Humedad Mínima	35,00
Humedad Máxima	65,00
Proporción de la Base	2,00
Proporción del Catalizador	1,70
Proporción del Diluyente	1,00

Figura 16: Actualización de datos estándar de medición

Como puede comprobarse en la figura 16 se crea y/o modifica el estándar sin presentar error además de haberse incluido la fecha y hora actual de creación/modificación. En caso de introducirse algún dato incorrecto al intentar guardar los cambios presentaría un error como el que se muestra en la siguiente figura 17.



The screenshot shows the 'Easy Box' web application interface. At the top, there is a navigation bar with 'Easy Box', 'Administrador', and 'Operario'. Below this, a 'Nuevo' button is visible next to 'Box Admin Action' and 'API 02.05.09 313-02'. A red box highlights the top section, and a black error message box states 'Unable to save. Correct the issue or discard changes'. Below the error message, the title 'API 02.05.09 313-02' is displayed. The main content area shows a table of parameters with the following data:

Parameter	Value
06/05/2024 11:38:08	
Temperatura Mínima	21,00
Temperatura Máxima	25,00
Tiempo de Reacción Mínimo	Ninguno
Tiempo de Reacción Máximo	0,00
Humedad Mínima	35,00
Humedad Máxima	65,00
Proporción de la Base	2,00
Proporción del Catalizador	1,70
Proporción del Diluyente	1,00

A red arrow points to the 'Ninguno' value in the 'Tiempo de Reacción Mínimo' row.

Figura 17: Prueba actualización de estándar error

## PU2. Introducción de datos de medición:

- *Descripción:* En esta prueba se comprobará que se satisface el requisito funcional 4 ([RF4](#)). Probaremos siendo un usuario operario a incluir nuevos datos de medición siendo conformes con la normativa la cual ha elegido mediante cuadro de selección y tiene como referencia en la misma vista sin poder modificarlo.
- *Precondiciones:* El administrador debe haber introducido la normativa de referencia y haber creado al usuario
- *Resultado esperado:* Tras incluir nuevos datos de medición, estos se deberán guardar sin presentar ningún error.
- *Resultado de la prueba:* Prueba pasada con éxito.

Como puede comprobarse en la figura 18 se crea y/o modifica la medida del operario Enrique Raposo López sin presentar error, además se incluye la fecha y hora actual de creación/modificación. En caso de introducirse algún dato incoherente con los datos estándar, al intentar guardar los cambios presentaría un warning que lo veremos en las pruebas de sistema.

Easy Box

Administrador

Operario

Nuevo

Box Operario Action

Enrique Raposo López

### Mediciones del Operario

Fecha de la mezcla

06/05/2024 12:05:43

Nombre

Enrique Raposo López

Número de Operario

Q8

Categoría Profesional

Especialista

Nivel de Operario

Nivel Superior

Normativa

AIPI 02.05.09 313-02

Temperatura

23,00

Humedad

55,00

Tiempo de Reacción

0,00

Proporción de la Base

2,00

Proporción del Catalizador

1,70

Proporción del diluyente

1,00

#### AIPI 02.05.09 313-02

Temperatura Mínima

21,00

Temperatura Máxima

25,00

Humedad Mínima

35,00

Humedad Máxima

65,00

Tiempo de Reacción Mínimo

0,00

Tiempo de Reacción Máximo

0,00

Proporción de la Base

2,00

Proporción del Catalizador

1,70

Proporción del Diluyente

1,00

Figura 18: Introducción de datos de medición

## 8.2. Pruebas de sistema

### PS1. Sistema de aviso de incompatibilidades o warning:

- *Descripción:* En esta prueba se comprobará que se satisface el requisito de alto nivel 5 ([RAN5](#) y [RF5](#)). Probaremos siendo un operario a incluir datos de medición siendo NO conformes con la normativa la cual ha elegido y veremos cómo se activa el sistema de warning.
- *Precondiciones:* El administrador debe haber introducido la normativa a referenciar incluyendo todas las medidas estándar a comparar.
- *Resultado esperado:* Una vez el operario ha introducido todos sus datos de medición, el sistema de warning hace un chequeo de cada medición haciendo una detección de la medida incompatible con los estándares vigentes y lanzando una ventana modal avisando al operario del campo incorrecto para su corrección.
- *Resultado de la prueba:* prueba pasada con éxito.

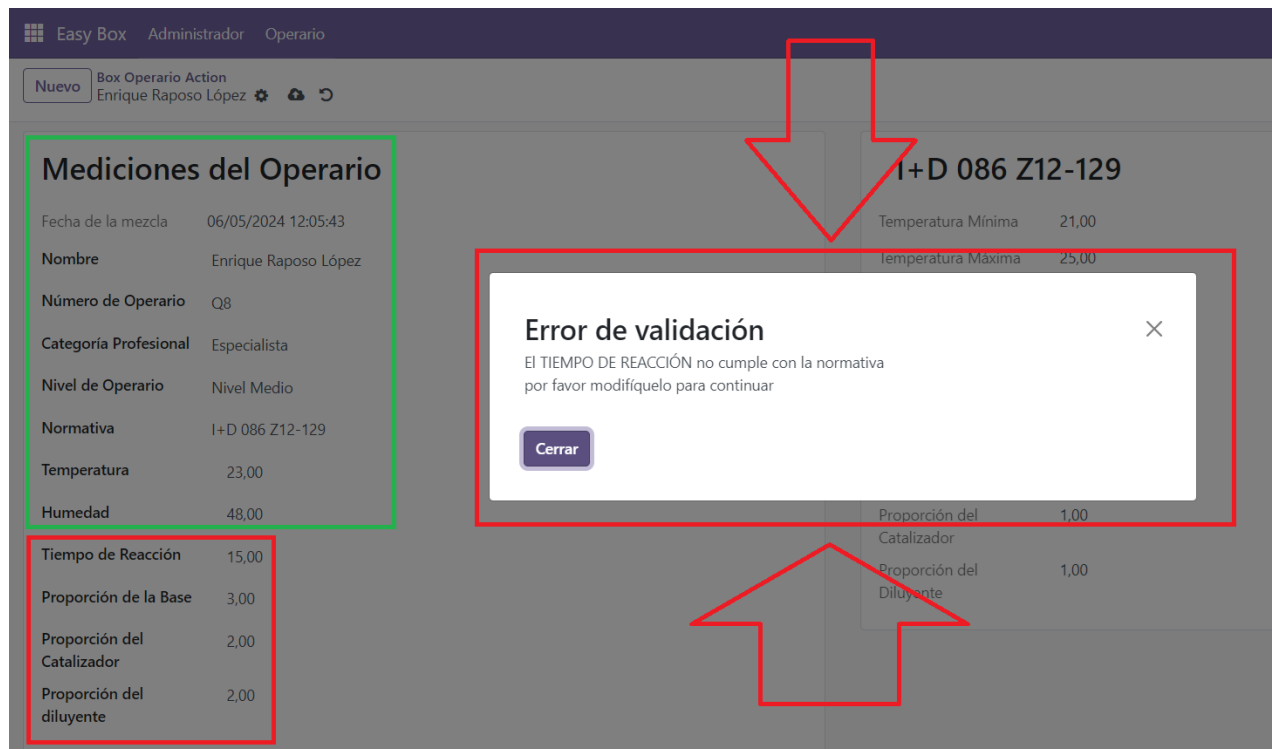


Figura 19: sistema Warning

Como se puede comprobar en la figura 19 el sistema hace un chequeo y en este caso la primera medida que detecta que no entra dentro del estándar elegido ("I+D Z12-129" en este caso) es el tiempo de reacción y avisa al operario para su modificación, una vez corregido podrá continuar guardando los cambios si ya son todos válidos con la normativa elegida vigente. Si fuesen todos correctos el sistema no hace ningún aviso y se podría guardar los datos de medición tal como se comprobó en la [PU2](#) "Introducción de datos de medición"

## 9. Conclusiones

### 9.1. Lecciones aprendidas

Lo que más he aprendido es a desarrollar código en lenguaje de programación Python ya que hasta ahora solo habíamos desarrollado básicamente en lenguaje java y en proyectos de pequeña escala por lo que además se ha aprendido a desarrollar un software completo (análisis, diseño, implementación, pruebas y su correspondiente documentación). A nivel técnico he aprendido en profundidad de las tecnologías usadas puesto que solo había trabajado con programas a nivel muy básico.

### 9.2. Objetivos cumplidos

**OBJ1:** Crear un nuevo módulo dentro del ERP Odoo con poder para crear perfiles a cada operario según categoría profesional y que Incluya datos relevantes de la normativa vigente en una base de datos que el operario debe conocer antes de la realización de la mezcla de

**componentes para la pintura:** Este objetivo se considera **cumplido** ya que se ha podido crear un nuevo módulo desde cero e incluirlo dentro de la estructura general de carpetas del SGE. En él se pueden crear operarios según categoría profesional y albergar datos del estándar normativo vigente visibles a cada operario en el momento de hacer una mezcla de pintura siendo este quien pueda elegir la normativa a seguir mediante un cuadro de selección igual que al elegir su nivel profesional con otro cuadro de selección y siendo el usuario administrador quien cree o modifique los datos estándar de la normativa.

**OBJ2: Crear una aplicación que avise de incompatibilidades con la normativa:** Este objetivo fundamental se considera **cumplido** al realizar la aplicación el sistema de warning convenientemente tal como se ha demostrado en la prueba de sistema 1 ([PS1](#))

**OBJ3: Crear un nuevo módulo de Odoo con una interfaz fácil e intuitiva para el operario:** Este objetivo está **cumplido** ya que nuestro módulo es de fácil aprendizaje y fácil manejo incluso por personal no cualificado.

### 9.3. Trabajo futuro

De cara al futuro se le podría agregar más funcionalidad:

- a) En cuanto a los datos de medición introducidos por cada operario sería conveniente poder reunirlos en una FR20 tal como se hace hoy día manualmente, pero en este caso también poder imprimirlo y que se pudiera crear desde nuestra aplicación solo para que el personal de control de calidad validara con su sello personal.
- b) Sería muy interesante poder incluir en el futuro una firma digital para que el operario firmara digitalmente su medición.
- c) Para un futuro sería bueno poder incluir más valores estándar para su validación como por ejemplo el número de lote de cada componente de la pintura que se va a usar, la fecha de caducidad de los mismos, tiempo a esperar entre cada mano de pintura, tiempo de evaporación antes del inicio del secado, tiempo de secado en estufa o a temperatura ambiente, etc.
- d) Otra de las mejoras que se podrían hacer es incluir otro menú en el que se pudiera crear un PNC rápidamente incluyendo fotografías de los defectos encontrados antes del comienzo del proceso de pintura para depurar responsabilidades.

En definitiva, el trabajo futuro de esta aplicación es inmenso, ya que hay muchísimos detalles que serían interesantes digitalizar durante el proceso de aplicación de la protección superficial.



## 10. Referencias bibliográficas

[1] "Odoo" (25/3/24)

- Odoo Community
- Documentación oficial Odoo ERP:  
<https://www.odoo.com/documentation/17.0/index.html>
- <https://vauxoo.github.io/odoo/howtos/backend.html>

[2] "Temario completo del ciclo formativo DAM" (14/2/24)

[3] "Software de Gestión de Calidad" <https://www.openinnova.es/odoo-software-gestion-calidad-libre/> (26-3-24)

[4] "Software en la nube con tecnología ISOTools para implementar y certificar Sistemas ISO" <https://www.kantansoftware.com/> (26-3-24)

[5] "Gestión Documental y Calidad en procesos altamente regulados" [https://www.marquesme.com/productos/verticales/shareme-gms-calidad/?20604153814&19550617096&qad\\_source=1&qclid=Cj0KCQjw2a6wBhCVARIsABPeH1tWgBu76FVSnVoOTM3PXeqmiec0SDTurQLq8vHUBsOXtXB\\_DXzajHgaAvuxEALw\\_wcB](https://www.marquesme.com/productos/verticales/shareme-gms-calidad/?20604153814&19550617096&qad_source=1&qclid=Cj0KCQjw2a6wBhCVARIsABPeH1tWgBu76FVSnVoOTM3PXeqmiec0SDTurQLq8vHUBsOXtXB_DXzajHgaAvuxEALw_wcB) (26-3-24)

[6] "Norma ISO 25010" <https://normasiso.org/norma-iso-25010/> (2/4/24)

[7] "Diagramas de casos de uso en la práctica" <https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/diagrama-de-casos-de-uso/> (6/4/24)

## 11. Anexos

### 11.1. Manual de usuario

Con este manual de usuario aprenderemos a hacer una instalación y uso del sistema Easy Box de una forma fácil y sencilla.

Una vez descargado e instalado el ERP Odoo en la sección “Fabricación” (así lo indicamos en el fichero \_\_manifest\_\_.py) de la pestaña aplicaciones aparecerá nuestro módulo recién creado, le damos a activar y ya lo tenemos disponible para ser usado. Hay que indicar que Odoo para no sobrecargar el sistema a partir de su versión 14 es necesario iniciarnos como superusuario para poder tener visible nuestro modulo activado en el desplegable de aplicaciones.

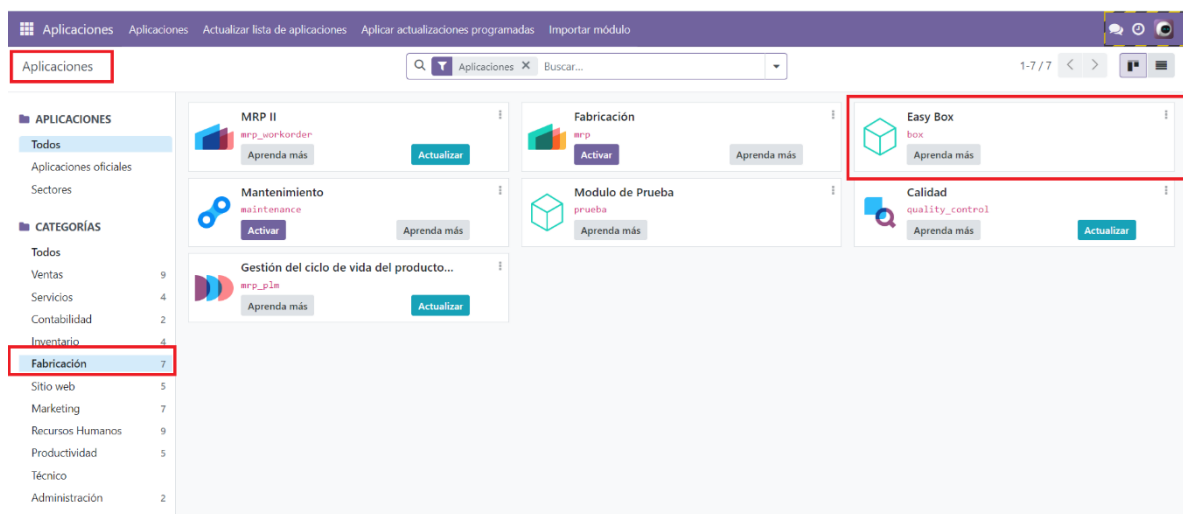


Figura 20: Aplicaciones Fabricación

Estar iniciado como superusuario se debe activar yendo a ajustes > activar el modo desarrollador y en el símbolo del desarrollador (el bichito) pinchar y dar a “convertirse en superusuario” con esto nuestra aplicación está activada y disponible en la lista



Figura 21: Lista aplicaciones instaladas

Al entrar en la aplicación nos aparecerán 2 pestañas en la parte superior, siendo la primera únicamente para el administrador donde puede agregar o modificar normativas.

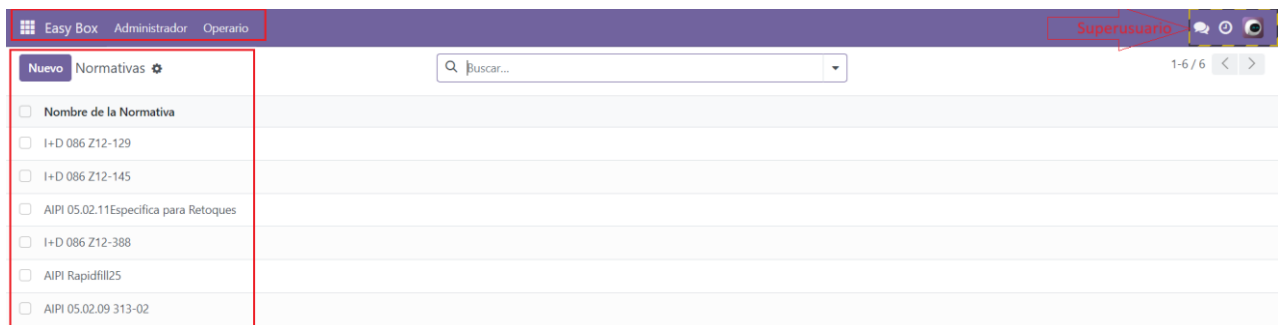


Figura 22: Menú administrador

Una vez en el menú administrador se puede crear un nuevo registro pulsando sobre el botón “Nuevo” o modificar algún registro ya existente simplemente pinchando encima del que se quiera modificar donde aparecerán todos los datos requeridos. Para guardar los cambios tanto si hemos creado un nuevo registro como si se ha modificado alguno, basta con darle al icono de la nube con una flecha y si todo es correcto se guardarán los cambios.

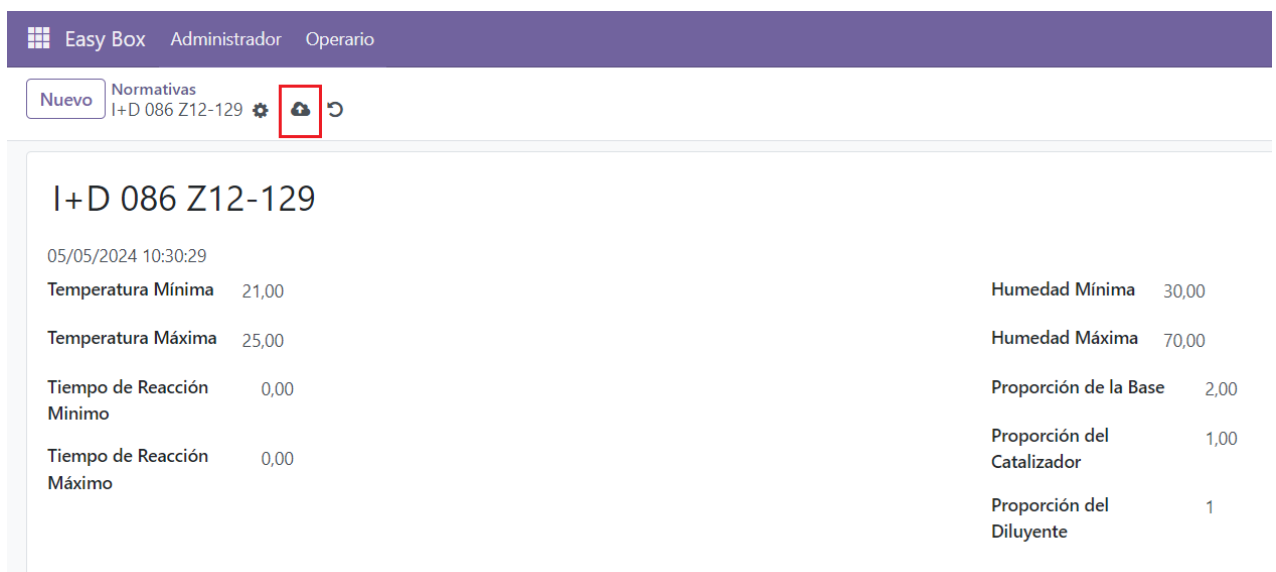


Figura 23: Detalles de un registro Normativa

La segunda de las pestañas del menú es para los operarios donde podrán introducir sus datos de medición y serán validados por el sistema en el momento de guardar los cambios pulsando igualmente en el icono de la nube con una flecha. Dentro de cada operario hay que introducir unos valores que son requeridos por el sistema siendo imposible continuar sin ser completados. En este apartado cabe destacar que tanto el dato requerido “Nivel de operario” y “Normativa” se podrán elegir de una lista desplegable.

The screenshot shows the 'Mediciones del Operario' form. The 'Nivel de Operario' dropdown menu is open, showing three options: 'Nivel Medio' (selected), 'Nivel Superior', and 'Nivel Básico'. The form includes fields for 'Fecha de la mezcla', 'Nombre', 'Número de Operario', 'Categoría Profesional', 'Temperatura', 'Humedad', 'Tiempo de Reacción', 'Proporción de la Base', 'Proporción del Catalizador', and 'Proporción del diluyente'. To the right, there is a table titled 'I+D 086 Z12-129' with various parameters and their values.

I+D 086 Z12-129	
Temperatura Mínima	21,00
Temperatura Máxima	25,00
Humedad Mínima	30,00
Humedad Máxima	70,00
Tiempo de Reacción Mínimo	0,00
Tiempo de Reacción Máximo	0,00
Proporción de la Base	2,00
Proporción del Catalizador	1,00
Proporción del Diluyente	1,00

Figura 24: Elección nivel de operario

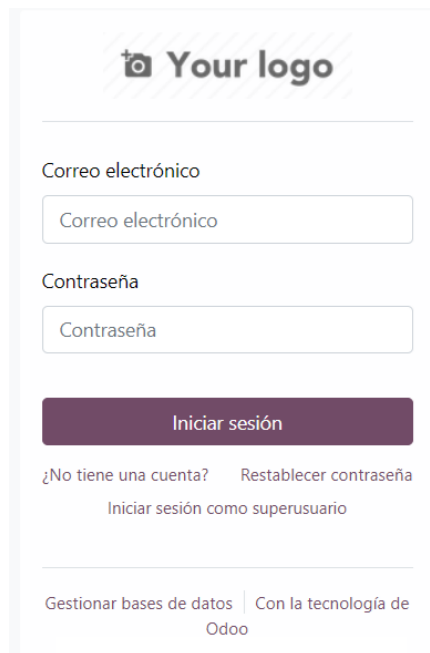
En función de la normativa que elija el operario se mostrarán en la parte derecha de la pantalla los datos de referencia sin poder ser estos modificados por el operario. Una vez introducidos todos los datos requeridos damos a guardar y si todo es correcto podemos finalizar la operación.

The screenshot shows the 'Mediciones del Operario' form. The 'Normativa' dropdown menu is open, showing a list of options including 'I+D 086 Z12-129' (selected), 'I+D 086 Z12-145', 'API 05.02.11Especifica para Retoques', 'I+D 086 Z12-388', 'API Rapidfill25', 'API 05.02.09 313-02', and 'Buscar más...'. A red arrow points from the 'Normativa' dropdown to the 'I+D 086 Z12-129' table on the right. The table contains various parameters and their values.

I+D 086 Z12-129	
Temperatura Mínima	21,00
Temperatura Máxima	25,00
Humedad Mínima	30,00
Humedad Máxima	70,00
Tiempo de Reacción Mínimo	0,00
Tiempo de Reacción Máximo	0,00
Proporción de la Base	2,00
Proporción del Catalizador	1,00
Proporción del Diluyente	1,00

Figura 25: Elección de normativa y muestra de referencia

Una vez queramos cerrar sesión en la parte superior derecha aparece el usuario que ha iniciado sesión, pinchamos en él y elegimos cerrar sesión. En caso de querer iniciar sesión simplemente introduce tus datos en a la pantalla de login proporcionada por el propio ERP y ya iniciarás sesión con el usuario predefinido por el administrador en la sección “ajustes > gestión de usuarios” con los permisos otorgados.



The login form features a header with a camera icon and the text "Your logo". Below this, there are two input fields: "Correo electrónico" and "Contraseña". A prominent purple button labeled "Iniciar sesión" is positioned below the password field. Underneath the button, there are links for "¿No tiene una cuenta?", "Restablecer contraseña", and "Iniciar sesión como superusuario". At the bottom, it says "Gestionar bases de datos" and "Con la tecnología de Odoo".

Figura 26: Inicio de sesión

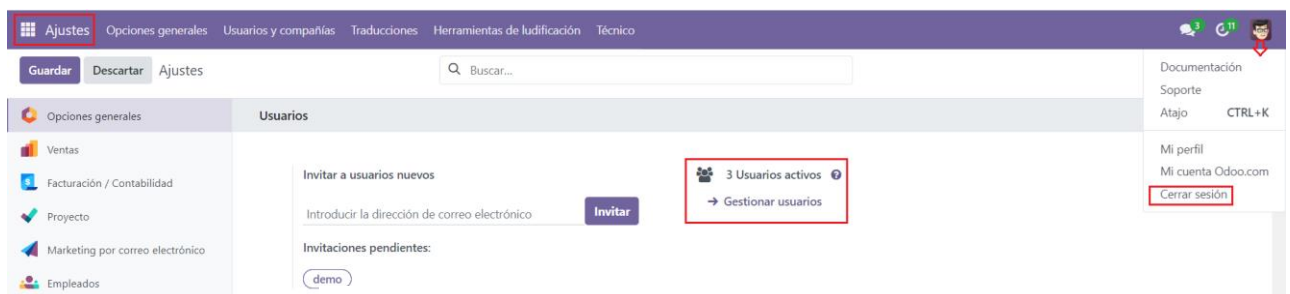


Figura 27: Gestión de usuarios/Cierre de sesión

Para concluir podemos ver en la siguiente figura 28 una información detallada del módulo que hemos creado al hacer click dentro del propio módulo en la parte superior derecha en los tres puntitos verticales y seleccionando “información del módulo”. Aparecerá lo mostrado en la siguiente figura.

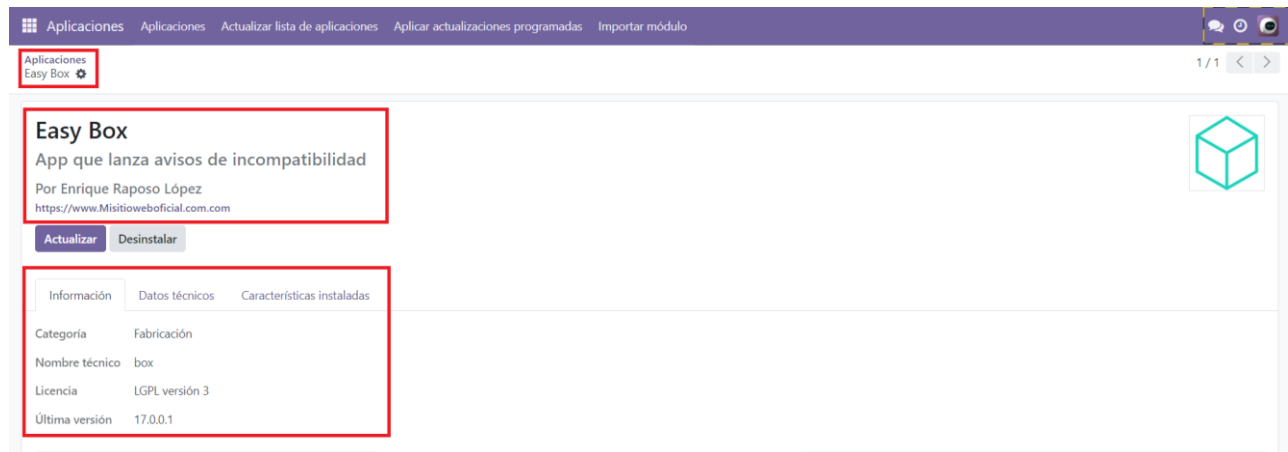


Figura 28: Información Easy Box

## 11.2. Porciones de código explicado

En este apartado vamos a mostrar porciones de código usados en la aplicación y se hará una breve explicación del mismo.

- La siguiente porción de código corresponde con la clase `BoxUsuario` perteneciente al fichero `models.py` y es muy interesante su explicación ya que en él se puede comprobar la estructura de un fichero Python ya que es un lenguaje orientado a objetos. Esta clase representa la tabla de una base de datos donde almacenaremos diferentes registros en campos tales como el Nombre del operario en el campo `name`, en `operario_number` se almacena el número de operario entre otros, los cuales usarán el sistema de ORM de `odoo` para almacenar los objetos creados en las tablas de la base de datos. Uno de los más interesante en cuanto a su sintaxis es el campo `fecha_hora_mezcla` ya que le añadimos el atributo "default" para que almacene un dato por defecto, en este caso almacenaremos por defecto la fecha y hora que recibe desde la función "write". Este campo lo definimos desde la vista como de solo lectura para que el operario no pueda introducirlo manualmente y poder hacer "trampas" y así cumplir cierta parte de la normativa sin ser cierta.
- Otra de la función altamente interesante es la función que lanza los avisos de incompatibilidad llamada "`_check_valores_normativa`" la cual recibe como parámetros los valores introducidos por el operario y los compara con los introducidos por el administrador en la clase llamada "`BoxNormativa`" lanzando el aviso de incompatibilidad si los valores no están dentro del rango permitido. Mediante un bucle "for" irá recorriendo cada campo y comparándolo mediante el condicional "if" cada valor introducido con el de referencia.

- También es importante hacer mención a la relación entre tablas mediante la sentencia “normativa\_id = fields.Many2one('box.normativa', string='Normativa', required=True)” ya que esta permite al operario seleccionar la normativa con la cual debe hacer comparación el sistema warning.

models.py

```
class BoxUsuario(models.Model):
    _name = 'box.usuario'
    _description = 'Usuario para Easy Box'

    name = fields.Char(string='Nombre', required=True)
    operario_number = fields.Char(string='Número de Operario', required=True)
    categoria_profesional = fields.Char(string='Categoría Profesional',
required=True)
    fecha_horamezcla = fields.Datetime(string='Fecha de la mezcla',
default=fields.Datetime.now)
    nivel_profesional = fields.Selection([('S', 'Nivel Superior'), ('M', 'Nivel
Medio'), ('B', 'Nivel Básico')], string='Nivel de Operario',
required=True)
    temp = fields.Float(string='Temperatura', required=True)
    humedad = fields.Float(string='Humedad', required=True)
    tiempo_reaccion = fields.Float(string='Tiempo de Reacción', required=True)
    proporcion_base = fields.Float(string='Proporción de la Base', required=True)
    proporcion_catalizador = fields.Float(string='Proporción del Catalizador',
required=True)
    proporcion_diluyente = fields.Float(string='Proporción del diluyente',
required=True)

#relación muchos a uno para poder elegir de entre varias normativas creadas
    normativa_id = fields.Many2one('box.normativa', string='Normativa',
required=True)

# Campos de referencia de la normativa. Mostrará las medidas estándar que el
administrador haya introducido
    name_normativa = fields.Char(string='Nombre de la Normativa',
related='normativa_id.name')
    temp_min_normativa = fields.Float(string='Temperatura Mínima',
related='normativa_id.temp_min')
    temp_max_normativa = fields.Float(string='Temperatura Máxima',
related='normativa_id.temp_max')
    humedad_min_normativa = fields.Float(string='Humedad Mínima',
related='normativa_id.humedad_min')
    humedad_max_normativa = fields.Float(string='Humedad Máxima',
related='normativa_id.humedad_max')
    tiempo_reaccion_min_normativa = fields.Float(string='Tiempo de Reacción
Minimo', related='normativa_id.tiempo_reaccion_min')
```



```
tiempo_reaccion_max_normativa = fields.Float(string='Tiempo de Reacción
Máximo', related='normativa_id.tiempo_reaccion_max')
proporcion_Base_normativa = fields.Float(string='Proporción de la Base',
related='normativa_id.proporcion_Base')
proporcion_Catalizador_normativa = fields.Float(string='Proporción del
Catalizador', related='normativa_id.proporcion_Catalizador')
proporcion_Diluyente_normativa = fields.Float(string='Proporción del
Diluyente', related='normativa_id.proporcion_Diluyente')

#función para actualizar la fecha y hora en la que se INTRODUCEN MEDICIONES y/o al
modificar cualquier registro
def write(self, values):
    if 'fecha_hora_mezcla' not in values:
        values['fecha_hora_mezcla'] = fields.Datetime.now()
    return super(BoxUsuario, self).write(values)

#función para la detección de incompatibilidades
@api.constrains('temp', 'humedad', 'tiempo_reaccion', 'proporcion_base',
'proporcion_catalizador', 'proporcion_diluyente')
def _check_valores_normativa(self):
    for user in self:
        normativa = user.normativa_id # Obtiene la normativa seleccionada por
el operario
        if normativa:
            if user.temp < normativa.temp_min or user.temp >
normativa.temp_max:
                raise exceptions.ValidationError('La TEMPERATURA no cumple con
la normativa\npor favor modifíquela para continuar')
            if user.humedad < normativa.humedad_min or user.humedad >
normativa.humedad_max:
                raise exceptions.ValidationError('La HUMEDAD no cumple con la
normativa\npor favor modifíquela para continuar')
            if user.tiempo_reaccion < normativa.tiempo_reaccion_min or
user.tiempo_reaccion > normativa.tiempo_reaccion_max:
                raise exceptions.ValidationError('El TIEMPO DE REACCIÓN no
cumple con la normativa\npor favor modifíquelo para continuar')
            if user.proporcion_base < normativa.proporcion_Base or
user.proporcion_base > normativa.proporcion_Base:
                raise exceptions.ValidationError('La proporción de la BASE no
coincide con la normativa\npor favor modifíquela para continuar')
            if user.proporcion_catalizador < normativa.proporcion_Catalizador
or user.proporcion_catalizador > normativa.proporcion_Catalizador:
                raise exceptions.ValidationError('La proporción del CATALIZADOR
no coincide con la normativa\npor favor modifíquela para continuar')
            if user.proporcion_diluyente < normativa.proporcion_Diluyente or
user.proporcion_diluyente > normativa.proporcion_Diluyente:
                raise exceptions.ValidationError('La proporción del DILUYENTE
no coincide con la normativa\npor favor modifíquela para continuar')
```

Este sería en cuanto al modelo de la aplicación que hace referencia a la base de datos donde se almacenarán los valores.

Por último, en cuanto al funcionamiento de la aplicación y no por ello menos importante nos encontramos con las vistas las cuales están codificadas mediante XML:

views.xml

```
<odoo>
<data>

  <!-- VISTA DE USUARIO -->
  <record id="view_box_user_form" model="ir.ui.view">
    <field name="name">box.user.form</field>
    <field name="model">box.usuario</field>
    <field name="arch" type="xml">
      <form string="Datos del Usuario">

<!-- Campos de referencia de la normativa -->
        <sheet>
          <group>
            <h1><field name="normativa_id" invisible="1"/></h1>
            <h1><field name="name_normativa" readonly="1"/></h1>
          </group>

          <group>
            <field name="temp_min_normativa" readonly="1"/>
            <field name="temp_max_normativa" readonly="1"/>
            <field name="humedad_min_normativa" readonly="1"/>
            <field name="humedad_max_normativa" readonly="1"/>
            <field name="tiempo_reaccion_min_normativa" readonly="1"/>
            <field name="tiempo_reaccion_max_normativa" readonly="1"/>
            <field name="proporcion_Base_normativa" readonly="1"/>
            <field name="proporcion_Catalizador_normativa" readonly="1"/>
            <field name="proporcion_Diluyente_normativa" readonly="1"/>
          </group>
        </sheet>

      </form>
    </field>
  </record>
```

```
<!-- DEFINICIÓN DE LOS MENÚS -->

<!-- Definición de Acciones -->
<record model="ir.actions.act_window" id="box.action_admin">
  <field name="name">Normativas</field>
  <field name="res_model">box.normativa</field>
  <field name="view_mode">tree,form</field>
</record>

<record model="ir.actions.act_window" id="box.action_operario">
  <field name="name">Operarios</field>
  <field name="res_model">box.usuario</field>
  <field name="view_mode">tree,form</field>
</record>

<!-- Top menu item -->
<menuitem name="Easy Box" id="menu_box_root" sequence="10"/>

<!-- Menu categories -->
<menuitem name="Administrador" id="menu_box_admin" parent="menu_box_root"
sequence="10" action="box.action_admin"/>
<menuitem name="Operario" id="menu_box_operario" parent="menu_box_root"
sequence="20" action="box.action_operario"/>
<menuitem name="Normativa" id="menu_box_normativa" parent="menu_box_root"
sequence="30"/>

</data>
</odoo>
```

En este caso nos quedamos con la vista del operario que es el más relevante. En esta porción de código se puede hacer interesante hacer mención a la definición de los menús. Los menús son los nombres o pestañas que aparecen en la parte superior de la pantalla que al clicar sobre ellos desencadena una acción definida que lanzará al controlador y nos mostrará por pantalla lo contenido en el modelo correspondiente.

En cuanto a la vista del operario hay que hacer mención que el operario tendrá la opción de elegir desde un menú desplegable la normativa con la que debe hacer la mezcla al igual que debe elegir de un menú desplegable su nivel de operario según sea básico, medio o superior.