UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escola Tècnica Superior d'Enginyeria Informàtica

etsinf

Escola Tècnica Superior d'Enginyeria Informática
Universitat Politècnica de Valéncia

Despliegue y Ampliación del ERP Odoo para un Negocio de Panadería Tradicional

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

Autor: Mario Luna Canet

Tutor: José Vicente Busquets Mataix

Curso 2021-2022

Resumen

Este proyecto consiste en el análisis y el desarrollo de un sistema para la gestión total de un modelo de negocio de panadería tradicional. El objetivo de este despliegue es agilizar los procesos que actualmente se hacen manualmente mediante la automatización y quitar carga al personal que se encarga de estos procesos, además de dotarlos de una plataforma para pedidos online. La empresa es una panadería tradicional la cual sirve en grandes cantidades tanto al público como a bares y restaurantes. En la actualidad la mayoría del papeleo y transmisión de información interna se hace de forma manual, lo que ralentiza mucho varios procesos, como los pedidos telefónicos o la organización del reparto del día siguiente y hace que necesiten una constante revisión. Como se ha mencionado con anterioridad también se encuentra en la necesidad de una pequeña página web que sea accesible solo para los clientes (restaurantes) para que realicen los pedidos diarios, así como un control de asistencia para los empleados.

Todas estas necesidades serán cubiertas con la herramienta ERP Odoo, en su versión Community, la cual ofrece un amplio catálogo de módulos gratuitos que cubren muchas de las funcionalidades necesarias para la empresa, desde el control de asistencia para empleados hasta la creación de la web de venta. La herramienta será desplegada mediante la utilización de Docker. Docker permite tener varias instancias corriendo con distintas versiones de Odoo, lo que hace mucho más sencillo el desarrollo y la activación del programa en cualquier momento y en cualquier entorno.

Para completar más la herramienta y adaptarla a las necesidades de la empresa se realizarán ampliaciones en varios de los módulos. En primer lugar, en el módulo *Contactos* se crearán nuevos campos para recabar información más específica del tipo de negocio y los horarios de reparto los cuales luego servirán para crear un documento el cual se rellenará con la información recabada en estos campos. Y por último al módulo de *Empleados* se le dotará de la funcionalidad para generar un acuerdo de confidencialidad que proteja las recetas y secretos del negocio.

Palabras Clave

Panadería, Herramienta ERP, Página web, Odoo Community

Abstract

This project consists of the analysis and development of a system for the total management of a traditional bakery business model. The objective of this deployment is to streamline the processes that are currently done manually through automation and take the burden off the staff in charge of these processes, in addition to providing them with a platform for online ordering. The company is a traditional bakery that serves large quantities to the public as well as to bars and restaurants. Currently most of the paperwork and transmission of internal information is done manually, which slows down several processes, such as telephone orders or the organization of the next day's delivery, and requires constant review. As mentioned before, there is also the need for a small web page that is accessible only to customers (restaurants) to place daily orders, as well as an attendance control for employees.

All these needs will be covered with the ERP tool Odoo, in its Community version, which offers a wide range of free modules that cover many of the functionalities needed by the company, from the attendance control for employees to the creation of the sales website. The tool will be deployed using Docker. Docker allows to have several instances running with different versions of Odoo, which makes it much easier to develop and activate the program at any time and in any environment.

To further complete the tool and adapt it to the needs of the company, extensions will be made to several of the modules. Firstly, in the Contacts module new fields will be created to collect more specific information on the type of business and delivery schedules which will then be used to create a document which will be filled in with the information collected in these fields. And finally, the Employees module will be provided with the functionality to generate a confidentiality agreement to protect the recipes and secrets of the business.

Key words:

Bakery, ERP Tool, Website, Odoo Community

Índice de contenidos

Índice de tablas

Índice de imágenes

1. Introducción

El primer apartado de esta memoria es la introducción, que se encuentra dividida en tres partes: motivación, objetivos y estructura de la memoria. En la primera parte, motivación, se explicará porqué se ha elegido realizar este proyecto tanto desde el punto de vista personal como del tecnológico. En segundo lugar, en objetivos se mostrarán cuáles son los objetivos a conseguir con este proyecto. Y por último la estructura de la memoria en la que se muestran y explican todos los apartados que va a contener esta memoria.

1.1. Motivación

Hoy en día la sistematización informática de las empresas es clave para poder optimizar todos los recursos al máximo y reducir costes y más todavía en la situación de mercado que nos encontramos en la cual la pandemia se ha llevado por delante un poco más de 207.000 empresas en España. La aparición de tecnologías como las herramientas ERP será de gran ayuda para optimizar y acelerar el crecimiento de las empresas y aumentar su eficiencia haciendo más fácil no solo sobrevivir, sino crecer en estos tiempos inciertos en los que vivimos. Muchos pequeños negocios siguen llevando la gestión de las distintas partes de la empresa de forma rudimentaria, mediante estas herramientas ERP, en concreto Odoo, se podrán centralizar todos los distintos aspectos del negocio en un solo lugar y así poder automatizar los procesos mencionados con anterioridad que antes requerían tiempo. Por eso he decidido seleccionar este proyecto debido a que considero estas herramientas ERP como el futuro de la gestión de las pequeñas empresas.

Debido a las prácticas realizadas y a la experiencia adquirida durante las mismas se ha podido descubrir Odoo y observar tanto su eficiencia como el sin fin de posibilidades que presentan sus módulos y la posibilidad de crear propios o modificar los ya existentes para adaptarlos a nuestras necesidades específicas. La gestión de estos tipos negocios es algo que me es cercano ya que mi familia cuenta con una panadería tradicional, en el que nosotros realizamos toda la gestión de la empresa de forma rudimentaria, lo que nos conlleva un gran consumo de tiempo.

Por estas dos razones he decidido utilizar esta tecnología para aplicarla al tipo de negocio de una panadería, ya que no solo nos ahorraría tiempo de formas increíbles, sino que además nos permitirá expandirnos y mejorar los servicios ofrecidos a nuestros actuales y futuros clientes.

1.2. Objetivos

Lo que se pretende con este proyecto es desplegar y modificar la herramienta ERP Odoo para adaptarla a las necesidades de una panadería tradicional. Se creará una disposición de Odoo que contenga los módulos necesarios para cubrir las necesidades de este tipo de negocios y se realizarán modificaciones sobre dos de ellos para darles una funcionalidad extra que se adapte a las necesidades del negocio.

Para conseguirlo, debemos entender que son los softwares ERP, sobre todo el utilizado, Odoo en su versión Community. Se establecerá una instalación de este programa mediante la herramienta de despliegue Docker, sobre la cual se aportará también información además de indicar sus ventajas a la hora de utilizarse para desplegar servicios. También se deberá investigar y encontrar un conjunto de módulos que se adapten a las necesidades de este tipo de negocio, lo que incluirá una web donde los clientes puedan realizar los pedidos de alimentos que deseen y una aplicación de asistencia para que los empleados puedan fichar a la hora de comenzar su turno. Una vez se tenga una instalación completa y funcional pasarán a realizarse una serie de desarrollos para añadir funcionalidades extra a dos de los módulos. Estas funcionalidades harán posible en primer lugar generar un documento mediante un smart button el cual contendrá dirección, datos de contacto y horario de reparto, el cual varía diariamente, de los clientes de la panadería para agilizar su reparto por el personal y en segundo lugar añadiremos la funcionalidad de generar un contrato de privacidad con los datos de cada empleado listos para su firma.

1.3. Estructura de la memoria

En este apartado de la memoria se podrán encontrar enumerados los 9 capítulos que formaran la memoria y a su lado una pequeña descripción sobre que se comentará en cada uno de ellos.

1. **Introducción:** es el apartado en el cual nos encontramos ahora, en el cual se expone la motivación, los objetivos y la estructura de este TFG.
2. **Estado del arte:** en este segundo apartado se explica que son las herramientas ERP, de donde vienen, que son y cuáles son los principales ERP que existen actualmente. Por último, nos centraremos en la ERP que vamos a utilizar para este proyecto, Odoo y se comentarán los aspectos más relevantes del mismo, así como su funcionamiento. Además, se comenta la herramienta de despliegue Docker con la que se ha realizado la instalación.
3. **Análisis del problema:** en este apartado se enumerarán y comentarán cuales son las necesidades de la panadería además y por qué son necesarias las mejoras realizadas.
4. **Diseño de la solución:** en este apartado se mostrarán y describirán los módulos que conforman la instalación adaptada al tipo de negocio. Además, se diseñarán unos bocetos que muestren como será el diseño del punto de venta y que muestren como quedarán las vistas con nuestras modificaciones y ampliaciones.
5. **Desarrollo de la solución:** se procede a explicar cómo se ha realizado el proyecto, se comentarán tanto los aspectos técnicos que conforman los modelos y las vistas de Odoo como la herencia, que es una parte importante en el desarrollo de las ampliaciones. También se explicará sistemáticamente como se han realizado estas modificaciones y ampliaciones sobre los módulos seleccionados. Y por último como se ha modificado el sistema de Odoo para dejarlo listo y que se adapte a nuestras necesidades.
6. **Implementación:** en este apartado se explicará cómo se ha realizado la instalación en Odoo y se comentará la herramienta que se usará para su despliegue, Docker. Explicando que es y cómo funciona se intentará dar un mejor entendimiento de el por qué se ha elegido está herramienta para el despliegue.

7. **Pruebas:** se realizan pruebas de funcionalidad de los módulos seleccionados, así como una prueba de pedido en el punto de venta. Además, se mostrará como se crearán tanto el contrato de confidencialidad para los empleados como el documento con el horario de reparto para cada negocio.
8. **Conclusiones:** en este último apartado se comentarán si los objetivos propuestos han sido cumplidos y que problemas han ido apareciendo durante el desarrollo del proyecto.

2. Estado del arte

En este segundo apartado de la memoria se comentará la tecnología utilizad para la realización del proyecto. Comentaremos que son los ERP, de donde vienen, cuáles son sus principales características y cuáles son los principales ERP que podemos encontrar en el mercado actual. Posteriormente nos centraremos en el ERP que hemos decidido utilizar, Odoo comentando un poco sobre las ediciones existentes y su historia y concluiremos comentando su arquitectura.

2.1. El Origen de los ERP

Las herramientas ERP, cuyo significado es Enterprise Resource Planning son en esencia sistemas de gestión empresarial. Estas herramientas sirven para que las empresas puedan integrar y automatizar los procesos más importantes juntos con los datos empresariales en una misma plataforma, haciendo muchos más sencillo la gestión de los datos y de la empresa.

Podemos pensar que estas herramientas son un invento reciente, pero en realidad sus inicios datan de la década de 1940. Al final de la Segunda Guerra Mundial, debido a la necesidad del ejército de los Estados Unidos de gestionar su material bélico, se empezaron a utilizar los programas informáticos que más tarde serían considerados los precursores de los ERP. En esta época las instituciones militares eran las únicas que disponían de computadoras por lo tanto su uso quedaba restringido al uso militar.

Esto cambió en la década de 1960 cuando empezaron a aparecer las primeras computadoras empresariales, lo que marco el inicio de una nueva forma de gestionar la información en las empresas. En los comienzos habitualmente el software básico venía incluido con la compra del hardware, aunque también se podía contratar por encargo desarrollos a medida para que se adaptaran a las necesidades específicas de esa empresa. De este modo fue como llegaron a las empresas las primeras aplicaciones básicas conocidas como BOM (Lista de Materiales) y los softwares más sofisticados conocidos como IMC (Gestión y Control de Inventarios), los cuales adaptaban las herramientas de organización desarrolladas por los militares durante la década anterior al mundo civil.

Durante la década de 1970 hicieron su aparición los programas MRP (Planificación de Necesidades de Materiales) de mano de la compañía IBM. A diferencia de sus predecesoras, estas aplicaciones podrían controlar no sólo donde y de qué manera se usaban los materiales, sino que también podían prever cuando iba a ser necesarios reponerlos y en qué cantidad. Debido a esto los MRP son considerados los predecesores más directos de los ERP. También cada destacar que durante esta década fueron fundadas la mayoría de las empresas que hoy en día aun proveen software ERP, como SAP, fundada en 1972, Oracle y J.D. Edwards, en 1977 o Baan, en 1978.

En la década de 1980 los programas usados por las empresas para la planificación de su producción empezaron a evolucionar y a incluir otros ámbitos de la empresa a parte de las materias primas. De esta forma evolucionaron y pasaron a llamarse MRP-II (Planificación de Recursos de Producción) y comenzaron a introducir algunos de los aspectos financieros de la empresa como el coste de materias primas o los costes logísticos.

Finalmente llegamos a la década de 1990 en la cual nace lo que conocemos actualmente como ERP. La consultora Gartner fue la que acuñó el término ERP (Sistema de Planificación de Recursos Empresariales) para denominar a los nuevos programas de organización empresarial que estaban llegando al mercado y que poseían un alcance que superaba ampliamente los ámbitos de fabricación y finanzas, así que ya no tenía sentido que siguieran llamándose MRP. Según este nuevo enfoque un ERP era un sistema de información que podía respaldar decisiones tomadas en todas las áreas de la compañía. De esta forma pasó a convertirse en un software apto para cualquier tipo de negocio (1).

2.2. ¿Que son los ERP?

Después de conocer la historia de la aparición de los ERP vamos a ver cuáles son sus objetivos y características principales.

Las herramientas ERP son sistemas que administran la información de del negocio y además sirven para automatizar numerosas tareas de la empresa relacionados con los aspectos productivos u operativos de esta. Su objetivo principal es conseguir que todos los datos de la empresa estén conectado e integrados. Esto lo consiguen gracias a que están compuestos por módulos, los cuales aportan cada uno una funcionalidad diferente para cubrir todas las necesidades de las empresas. Las principales ventajas de los ERP son (2):

- Nos permite rentabilizar procesos, planificando tiempos y tareas para que los empleados sean lo más efectivos posible.
- Son completamente personalizables y adaptables a las necesidades de cualquier empresa
- Mejora en gran medida la comunicación interna entre distintos departamentos.
- Acceso total y sencillo a cualquier información de la empresa necesaria.

Los beneficios que pueden llegar a aportar las herramientas ERP se resumen en la resolución de los problemas fiscales, mercantiles y contables de la empresa. Lo que distingue a los ERP de cualquier otro software empresarial es que los ERP son modulares, configurables y especializados (2):

- **Modulares:** para los ERP las empresas son un conjunto de departamentos interrelacionados entre sí por la información que comparten y que generan sus procesos. Una ventaja técnica y económica de los ERP es que las funcionalidades se dividen en módulos porque lo que se puede obtener o instalar solo lo necesario.
- **Configurables:** mediante la realización de desarrollos sobre su código los ERP pueden ser configurados según las necesidades específicas de cada usuario.
- **Especializados:** los ERP especializados brindan soluciones existentes en áreas de mucha complejidad y bajo una estructura de constante evolución. Para las empresas que requieren soluciones reales para sus problemas específicos los ERP especializados son mucho más factibles ya que los ERP genéricos solo ofrecen soluciones generales que requieren desarrollos para cubrir estas necesidades específicas.

Dependiendo de la magnitud de la empresa los sistemas ERP pueden ser costosos y difíciles de implementar, ya que en muchos de los casos es necesario realizar una personalización que se adapte a las necesidades del cliente. Ya que la configuración genética de estos ERP es muy básica, las empresas necesitarán dedicar un tiempo de desarrollo para personalizar estás herramientas para satisfacer sus necesidades. A la hora de implantar estos sistemas, debido a la complejidad del proceso y del mantenimiento, las empresas suelen buscar la ayuda de un proveedor de ERP o de una consultora tecnológicas.

Como hemos visto anteriormente la implementación de una herramienta de este tipo también tiene sus desventajas e inconvenientes, que hay que tener en cuenta. Las que caben destacar son las siguientes (3):

- Requiere un tiempo de diseño e instalación: cuando se realiza un despliegue específico para una empresa se necesita un tiempo previo para diseñarlo.
- Necesita ser actualizado.
- Requiere un equipo físico para funcionar.
- Algunos necesitan estar conectados a Internet.
- Pueden tener costes fijos o periódicos que hay que contemplar.

2.3. Los principales ERP

Ahora pasaremos a exponer y explicar brevemente los principales ERP que se encuentran actualmente en el mercado, pero primero vamos a dejar claro unos conceptos básicos. En primer lugar, comentaremos los dos tipos de ERP que se pueden encontrar actualmente: los horizontales y los verticales (4).

Los ERP horizontales o generalistas son programas de gestión que cubren las funcionalidades de gestión de cualquier empresa, concretamente la gestión comercial y contable. Cubren las necesidades básicas genéricas de cualquier tipo de negocio, pueden ser personalizables y sus costes suelen ser bajos o incluso nulos.

Por otro lado, tenemos los ERP verticales o también llamados verticales o sectorizados, los cuales son programas de gestión diseñados para un sector especifico, por lo tanto incluyen funcionalidades específicas que no suelen encontrarse disponibles para los ERP horizontales. Este tipo se caracteriza porque aunque posea los módulos básicos de cualquier ERP, también incluye algún módulo que añade funcionalidades específicas que solo son útiles en ese sector en concreto.

En segundo lugar, hay que remarcar que dependiendo del tipo de instalación que se necesite realizar podemos encontrar tres tipos de este software:

- **Instalación en la nube:** cuando se elige esta opción lo que se hace en contratar los servicios de una empresa externa para que almacene tanto la parte del software como los datos de la empresa. Con este tipo de instalación lo único que nos hace falta es una conexión a internet para poder utilizar el servicio.
- **Software as a Service:** de las siglas SaaS que en español significa Software como servicio. De la misma manera que en la instalación en la nube contrataremos a una empresa externa un servicio por el cual obtendremos una solución software con el método de pago por uso, es decir como si alquilásemos su uso. Este método ha ido ganando fama año tras año debido a la gran ventaja que ofrece al dejar el mantenimiento y la gestión del software a una empresa externa.
- **Instalación local:** este tipo de instalación se realiza dentro del servidor y de la infraestructura de la empresa, por este motivo se debe tener en cuenta que se necesita poseer previamente una infraestructura capaz de alojar la instalación. Al ser una instalación en servidor propio será la propia empresa la responsable de la seguridad, la gestión y la disponibilidad del software.

Ahora que ya tenemos estos dos conceptos clave aclarados podemos pasar a mostrar los ERP más usados actualmente(5), destacando las características que los diferencian entre sí. Son los siguientes:

- **SAP ERP.** Una de las ERP más grandes y utilizadas por las pymes de todo el mundo. Fue desarrollada por la empresa alemana SAP (Sistemas de Análisis y Desarrollo del Programas) fundada en 1972. Este ERP se caracteriza por poseer una gran cantidad de procesos integrados que llegan a cubrir el 90% de los procesos de los negocios (5). También cabe destacar su instalación en la nube y su innovación en el campo del machine learning con la finalidad de realizar predicciones y mejores en la planificación del suministro, así como sugerencias de optimización. Por otra parte, este ERP es uno de los más caros del mercado y se requiere una formación mínima para utilizarlo ya que es un software complejo.
- **Oracle ERP.** Este software basado en la nube se utiliza para automatizar tanto las actividades comerciales cotidianas como los procesos administrativos. Este Paquete de software de gestión empresarial incluye(6): gestión financiera, gestión de la cadena de suministro, gestión de proyectos, contabilidad y compras. De esta forma, reuniendo todos los procesos empresariales, se consigue aumentar la colaboración y la productividad empresarial. Existen distintas versiones de este ERP en el mercado actualmente.
- **Microsoft Dynamics.** (7)En este ERP se distribuye de forma en que las empresas pagan por los servicios utilizados, es decir siguen el modelo se servicio SaaS. Es un servicio de aplicaciones empresariales que fusiona componentes de CRM y ERP, además de su compatibilidad con las otras herramientas de Microsoft. También hay que mencionar que se trata un ERP que posee una estructura horizontal como la descrita anteriormente, de manera que hay herramientas para tareas más específicas que no se encuentran disponibles.
- **Sage.** Hay que destacar su ERP para pymes, Sage 200 cloud. Como indica su nombre es un software basado en la nube (5). Destaca por su fácil escalabilidad y por su interfaz amigable, aunque siga requiriendo un tiempo para aprender a gestionarlo. El inconveniente de este ERP es que se trata de un ERP horizontal, por lo que es muy poco personalizable.

- **Workday.** Se trata de una herramienta de gestión de Recursos Humanos, que al igual que todos los ERP, tiene como objetivo automatizar procesos, centralizar toda la información, mejorar la toma de decisiones, entre otros. Destaca por centrarse en ofrecer soluciones cloud para la gestión integral de RRHH y Finanzas. Por último cabe mencionar que es una solución que se centra en compañías de gran tamaño, especialmente multinacionales (8).
- **Odoo.** Este es el ERP que he seleccionado para desarrollar este proyecto, debido en gran parte a su versión Community, la cual es gratuita, y que también se trata de un ERP de código abierto. Este ERP a grandes rasgos se podría considerar como un conjunto de aplicaciones que cuentan con unas herramientas de fácil uso enfocadas a rentabilizar y optimizar el negocio. En el siguiente apartado se comentará más en detalle este ERP.

2.4. El ERP seleccionado, Odoo

Desde 2005 Odoo ha ido creciendo constantemente. Este ERP es proporcionado por la empresa belga Odoo S.A. fundada en 2002 por Fabien Pickaers, su actual CEO. El primer ERP de esta compañía fue Tiny ERP el cual era un sistema modular el cual era configurable, escrito en Python(9) y usaba PostgreSQL(10)como base de datos. Tiny ERP fue su principal ERP desde 2005, cuando aparece, hasta 2009, cuando aparece su nueva herramienta. Este nuevo ERP se le llama OpenERP debido a su licencia libre y filosofía abierta, el cual se mantuvo hasta la aparición de Odoo en 2014.

Odoo cuenta actualmente con dos versiones: Odoo Enterprise y Odoo Community(11). En primer lugar comentaremos Odoo Enterprise. Esta versión se lanzó al mercado como la edición patentada de Odoo ERP, en la cual aparte de proporcionar una plataforma de código abierto y servicios de soporte, se ofrecía una ayuda a las empresas para optimizar aún más sus modelos comerciales mediante módulos y funciones extra con tarifas de licencia. En segundo lugar, se encuentra su versión Community, la cual he utilizado para la realización de este proyecto debido a las características que comentamos a continuación. Esta versión se caracteriza por ser una herramienta de código abierto la cual es totalmente gratuita, aparte de incluir de base una serie de módulos que sirven para cubrir las necesidades básicas de cualquier empresa.

La OCA (Asociación de la Comunidad de Odoo) es una organización sin ánimo de lucro cuya misión es promover el uso de Odoo y apoyar el desarrollo colectivo de nuevas herramientas para esta plataforma. Esta organización es uno de los pilares fundamentales de la versión Community ya que provee apoyo legal, organizativo y financiero a la comunidad de este ERP.

Odoo Community, como todo ERP, es modular lo que nos permite realizar la instalación únicamente de los módulos que realmente vamos a utilizar. Debido a que es un ERP muy extendido actualmente, esta versión posee una gran comunidad de desarrolladores, lo que es una gran ventaja ya que hay muchas ampliaciones que ya han sido realizadas por terceros, las cuales se pueden conseguir pagando o de manera gratuita.

En último, explicaremos las tecnologías antes mencionadas ya que son las tecnologías de las que hace uso Odoo: PostgreSQL, XML y Python. PostgreSQL, conocido también como Postgres, es un sistema relacional de gestión de bases de datos orientado a objetos, basado en SQL y de código abierto. XML son las siglas para Extensible Markup Language, este lenguaje es un metalenguaje que permite definir lenguajes de marcas. Este metalenguaje fue desarrollado por W3C y es utilizado para almacenar datos de forma legible, pero nosotros lo utilizaremos para para mostrar los nuevos campos y botones que añadirán las nuevas funcionalidades ya que las vistas de Odoo se crean mediante este lenguaje. Por último, hay que mencionar a Python que es un lenguaje de programación interpretado cuya ventaja principal es la legibilidad de su código. Es un lenguaje de programación multiparadigma porque soporta parcialmente la programación con orientación a objetos, la programación imperativa y también cabe mencionar sobre la programación funcional que es un lenguaje dinámico, multiplataforma e interpretado.

2.5. Funcionamiento de Odoo

A continuación, hablaremos sobre cómo está diseñado Odoo y su funcionamiento. Odoo es un software que sigue el patrón de arquitectura MVC (Modelo Vista Controlador), este modelo es multinivel y es utilizado por la mayoría de las aplicaciones informáticas. En esta arquitectura se encuentran diferenciados los tres niveles de la aplicación: presentación, lógica y datos.

- **Nivel de presentación (Vista).** También denominado interfaz de usuario es la capa de más alto nivel y es la responsable de mostrar los datos e interactuar con el usuario ya que es en sí misma la interfaz principal. Cualquier cosa que el usuario pueda ver o interactuar con ella se guarda en este nivel. En Odoo se nos proporciona mediante vistas sencillas e intuitivas el acceso a todos los distintos de departamentos de la empresa.
- **Nivel de lógica (Controlador).** Es el responsable de cada una de las interacciones entre el nivel de presentación y el nivel de datos además de encargarse de aplicar toda la lógica y cálculos necesarios. Para garantizar la consistencia y la seguridad de los datos en las bases de datos de nivel bajo se debe acceder a ellas únicamente mediante esta capa.
- **Nivel de datos (Modelo).** Es el modelo más bajo y se encarga de gestionar la información de la aplicación y asegurar su persistencia. En sí mismo representa la información con la que el sistema opera, debido a esto es el que se encarga de gestionar el acceso a dicha información. Odoo para realizar a cabo esta tarea se basa en un servidor PostgreSQL. Cualquier tipo de información como documentos o imágenes se guarda son almacenadas en los directorios del sistema. En este nivel se almacenará también la información y los datos que el nivel de lógica solicita para realizar sus operaciones.

3. Análisis del problema

En este apartado analizaremos el problema que se pretende resolver mediante este proyecto, para ello se seguirá el estándar IEEE 830 (12). En este estándar se muestra un conjunto de recomendaciones que sirven para poder especificar los requerimientos o necesidades que el software a desarrollar debe satisfacer. También mostraremos con unos bocetos los casos de uso que deben satisfacerse, tanto por parte del cliente externo que entra en la web como del interno que trabaja con el sistema.

3.1. Especificación de requisitos

Introducción

En este apartado se presentará una Especificación de Requisitos del Software, según el estándar IEEE 830. Este estándar también conocido como ERS, está formado por los subapartados siguientes: propósito, ámbito del sistema, definiciones y visión general del documento.

Propósito

El objetivo del ERS es establecer cuáles son los problemas a abarcar. Para conseguir un mejor entendimiento de cuales son estos problemas se detallará cada uno de ellos detenidamente, de este modo facilitaremos la implementación y el desarrollo de cada uno de ellos con el objetivo de que el producto final cubra todas las necesidades y exigencias del negocio.

Ámbito del sistema

En este proyecto se va a crear un despliegue de Odoo al que llamaremos Panificate, el cual será aplicado para una panadería tradicional, este despliegue permitirá a identificar y tener en cuenta a todos los clientes y los empleados y poder tener sus datos centralizados en un mismo lugar. Además, nos permitirá organizar una división de los turnos de trabajo y añadir un control de asistencia digital, que nos permita fichar con el móvil. También poseerá una interfaz web desde donde los restaurantes podrán encargar pan o pastelería de manera remota. Por último, debido a las modificaciones que realizaremos se podrá crear tanto un contrato de confidencialidad que se autocomplete con los datos del empleado como una hoja de reparto de los restaurantes añadiendo el horario en el que quieren recibir la comida.

Definiciones

Un gran número de términos presentados a continuación ya han sido definidos con anterioridad, los cuales son ERP, definido previamente en el apartado 2.2, MVC, en el 2.5 y Python, PostgreSQL y XML en el apartado 2.4.

Visión general de la ERS

Este subapartado sirve para comentar lo que se va a tratar en los siguientes apartados, la descripción general, en la cual se dará una visión de todos los factores que afectan a los requisitos del producto y al producto en sí. Además, se encuentran incluidos la funcionalidad del sistema, las características y permisos de los distintos grupos que accederán al sistema, las restricciones que contiene el propio sistema, las suposiciones y dependencias y en penúltimo lugar los requisitos futuros. El último de los apartados, el de los requisitos específicos, tiene como objetivo ofrecer una descripción lo detallada de los requisitos del sistema para que se pueda diseñar un sistema que cumpla con los requisitos demandados.

Descripción general

Perspectiva del producto

El despliegue de Odoo que se va a desarrollar no forma parte de un sistema más grande ni depende de ningún otro software para su funcionamiento, pero si a la hora de su implementación ya que usaremos Docker para su instalación y despliegue. De esta forma para un correcto despliegue y funcionamiento solo necesitaremos tener instalado Docker, Odoo en la versión deseada, en nuestro caso la 14, y el sistema de gestión de bases de datos PostgreSQL.

Funcionalidad del sistema

El objetivo principal es gestionar los clientes y los empleados de una panadería, además de añadir un comercio online para que se puedan realizar pedidos online. Las funciones que debemos concretar son:

- Gestionar empleados (dar de baja, crear)
- Crear contrato de confidencialidad
- Gestionar usuarios (clientes)
- Registro de Usuarios (clientes)

- Inicio de sesión de usuarios y empleados
- Realización de pedidos
- Crear PDF con horario de reparto
- Controlar las asistencias de los usuarios
- Almacenar registro de asistencias

Características de los usuarios

Esta aplicación va a dar sus servicios tanto a los empleados, como a los usuarios, como los administradores de la empresa. Los usuarios son los que menos permisos tendrán ya que únicamente podrán acceder a sus compras, a sus pedidos y su usuario para modificar los datos. De este modo entrarán en la web y podrán iniciar sesión y realizar sus compras.

En cuanto los empleados, tendrán acceso a al inicio de sesión y también al check in de asistencia, así podrán fichar sin necesidad de dispositivos externos y mediante el control de asistencias podremos identificar errores humanos y mejorar la eficiencia de los cambios de turno.

Y por último los Administradores que tendrán acceso total al sistema, esto conlleva poder ver y editar tanto los empleados como sus datos, lo mismo podrán hacer con todos los clientes que se registren en la web. Tendrán permisos de vista y edición en todos los módulos que formarán este despliegue. Además, se asignará un permiso único para los administradores sobre las nuevas funcionalidades que se añadirán para crear un contrato personalizado de confidencialidad y para crear un horario de reparto con todas las horas y direcciones de los clientes que en este caso serán bares o restaurantes.

Restricciones

El despliegue tiene ciertos requerimientos para su correcto funcionamiento y correspondiente testeo, son los siguientes:

- Máquina, física en nuestro caso, con una partición de Linux 20.04.
- Navegador web a nuestra elección (Firefox, Chrome), preferiblemente actualizado a su última versión.
- PostgreSQL instalado.

- Python con las librerías necesarias para las dependencias de Odoo-
- Una instalación correcta y funcional de Odoo 14 en el sistema.

Suposiciones y Dependencias

Este apartado se encuentra relacionado con el anterior, en la cual hemos indicado las versiones necesarias de los requisitos técnicos. La utilización de alguna versión diferente a las indicadas anteriormente puede variar el funcionamiento del sistema y la inutilización del mismo.

Requisitos futuros

Una vez se tenga la funcionalidad básica funcionando completamente se podrían realizar ampliaciones o incluso nuevas funcionalidades, como la posibilidad de que los empleados puedan postear ideas y sugerencias para la empresa o la implementación de un módulo que gestione la planificación de turnos de los empleados.

Requisitos específicos del sistema

Ahora se va a explicar cuáles son los requisitos específicos mostrados en formato tabla. Estos requisitos se centrarán en las principales funcionalidades que hemos planeado para los distintos grupos de usuarios, es decir para los usuarios de la web, los empleados y los administradores del negocio.

Cada tabla mostrar estos contenidos y seguirá la siguiente arquitectura:

Tipo de usuario:	Identificara a qué grupo de usuarios pertenece: Administrador, Cliente o Empleado.
Id de requisito:	Número que servirá para identificar el requisito.
Nombre del requisito:	Nombre el cual servirá para identificar el requisito.
Proceso realizado por el usuario:	Acciones realizadas por el usuario del grupo indicado
Proceso realizado por el sistema:	Acciones que realiza el sistema
Resultado de la interacción:	La situación que se debe dar al final de las interacciones entre usuario y sistema

Tabla 3.1. Tabla ejemplo para los requisitos.

Requisitos para el usuario de tipo Administrador

En primer lugar, mostraremos los requisitos específicos que el sistema debe cumplir sobre la funcionalidad que afecta a los usuarios tipo Administrador, estos requisitos los podemos ver claramente descritos en las tablas **3.2**, **3.3**, **3.4**, **3.5**, y **3.6**. Estos usuarios tendrán acceso total al sistema y aparte será los únicos que tengan la opción de utilizar las ampliaciones realizadas y de generar los nuevos documentos.

Tipo de usuario:	Administrador
Id de requisito:	01
Nombre del requisito:	Inicio de Sesión
Proceso realizado por el usuario:	El administrador iniciará sesión con su correo y contraseña.
Proceso realizado por el sistema:	El sistema identificará los permisos y le concederá los permisos de administrador.
Resultado de la interacción:	Se muestra la aplicación y se pueden observar todos los módulos instalados.

Tabla 3.2. Administrador – Inicio de Sesión

Tipo de usuario:	Administrador
Id de requisito:	02
Nombre del requisito:	Cierre de Sesión
Proceso realizado por el usuario:	El administrador clica en cerrar sesión.
Proceso realizado por el sistema:	El sistema lo reconoce y cierra la sesión del administrador.
Resultado de la interacción:	El administrador es redirigido a la página de inicio de sesión.

Tabla 3.3. Administrador – Cierre de sesión

Tipo de usuario:	Administrador
Id de requisito:	03
Nombre del requisito:	Acceso total al sistema
Proceso realizado por el usuario:	El administrador podrá acceder a todas las aplicaciones y tendrá permisos de edición sobre ellas.
Proceso realizado por el sistema:	Comprobar los permisos de visualización y edición del usuario y darle acceso a la totalidad del sistema.
Resultado de la interacción:	El administrador es capaz de visualizar y editar en todas las aplicaciones del sistema.

Tabla 3.4. Administrador – Acceso total al sistema

Tipo de usuario:	Administrador
Id de requisito:	04
Nombre del requisito:	Crear contrato de confidencialidad
Proceso realizado por el usuario:	El administrador clica en el botón de generar contrato que se encuentra dentro de cada empleado.
Proceso realizado por el sistema:	El sistema comprobará los datos del empleado y creará un contrato con los datos del empleado seleccionado.
Resultado de la interacción:	Se genera un documento PDF listo para ser imprimido y firmado por el empleado seleccionado.

Tabla 3.5. Administrador - Crear contrato de confidencialidad

Tipo de usuario:	Administrador
Id de requisito:	05
Nombre del requisito:	Crear documento con horarios de reparto
Proceso realizado por el usuario:	El administrador accederá al menú de los contactos, en nuestro caso clientes, y podrá generar un horario de reparto de los locales deseados.
Proceso realizado por el sistema:	Recopilará los horarios de reparto de los negocios seleccionados y generará un documento con la planificación diaria de reparto.
Resultado de la interacción:	Se generará el documento el cual podrá ser descargado.

Tabla 3.6. Administrador - Crear documento con horarios de reparto

Requisitos para el usuario de tipo Empleado

En segundo lugar, se mostrarán los requisitos específicos de funcionalidad que se requiere del sistema para los Empleados, los cuales los encontraremos claramente detallados en las tablas **3.7**, **3.8**, **3.9**. Estos requisitos se centrar mayormente en el inicio y cierre de sesión y en el check in de asistencia.

Tipo de usuario:	Empleado
Id de requisito:	06
Nombre del requisito:	Inicio de Sesión
Proceso realizado por el usuario:	El usuario introducirá sus credenciales para iniciar sesión.
Proceso realizado por el sistema:	El sistema reconocerá las credenciales e iniciará su sesión con los permisos de Empleado.
Resultado de la interacción:	Iniciará la sesión y redireccionará al Empleado al control de asistencia.

Tabla 3.7. Empleado - Inicio de Sesión

Tipo de usuario:	Empleado
Id de requisito:	07
Nombre del requisito:	Cierre de Sesión
Proceso realizado por el usuario:	El usuario clicará en cerrar sesión para salir del sistema.
Proceso realizado por el sistema:	Reconocerá al usuario y cerrará su sesión.
Resultado de la interacción:	Se cerrará la sesión del usuario y se mostrará la página de Inicio de Sesión.

Tabla 3.8. Empleado - Cierre de Sesión

Tipo de usuario:	Empleado
Id de requisito:	08
Nombre del requisito:	Check in de empleados.
Proceso realizado por el usuario:	Acciones realizadas por el usuario del grupo indicado
Proceso realizado por el sistema:	Acciones que realiza el sistema
Resultado de la interacción:	La situación que se debe dar al final de las interacciones entre usuario y sistema

Tabla 3.9. Empleado - Check in de empleados

Requisitos para el usuario de tipo Cliente

Por último, se mostrarán los requisitos específicos que afectan a los Clientes, los cuales serán Inicio de Sesión, en la tabla **3.10**, Cierre de Sesión, tabla **3.11**, Ajustes de Usuario, tabla **3.12**, y la realización del proceso de compra, Agregar al Carrito, en la **3.13**, y Comprar – Pago, en la tabla **3.14**.

Tipo de usuario:	Cliente
Id de requisito:	09
Nombre del requisito:	Inicio de Sesión
Proceso realizado por el usuario:	El cliente iniciará sesión con sus credenciales
Proceso realizado por el sistema:	El sistema reconocerá las credenciales y iniciará sesión.
Resultado de la interacción:	El cliente tendrá su sesión iniciada y se encontrará en la web de nuevo

Tabla 3.10. Cliente - Inicio de Sesión

Tipo de usuario:	Cliente
Id de requisito:	10
Nombre del requisito:	Cerrar Sesión
Proceso realizado por el usuario:	El usuario cerrará su Sesión.
Proceso realizado por el sistema:	El sistema reconocerá la acción y cerrará la sesión del cliente.
Resultado de la interacción:	Se cerrará la sesión y se redirigirá a la página.

Tabla 3.11. Cliente - Cerrar Sesión

Tipo de usuario:	Cliente
Id de requisito:	11
Nombre del requisito:	Editar los datos del propio usuario
Proceso realizado por el usuario:	El cliente entrará para editar sus propios datos.
Proceso realizado por el sistema:	El sistema les mostrará a los clientes sus datos actuales y le permitirá editar los existentes. Una vez hecho esto guardará los cambios en nuestra base de datos.
Resultado de la interacción:	Le mostrará al cliente sus cambios modificados y en su ficha de contacto se habrán modificado los campos también.

Tabla 3.12. Cliente - Editar los datos del propio usuario

Tipo de usuario:	Cliente
Id de requisito:	12
Nombre del requisito:	Agregar al carrito
Proceso realizado por el usuario:	El cliente agregará a su carrito los elementos que desee comprar
Proceso realizado por el sistema:	El sistema identificará el elemento y lo añadirá al carrito
Resultado de la interacción:	El producto será añadido al carrito y se podrá entrar para consultar

Tabla 3.13. Cliente - Agregar al carrito

Tipo de usuario:	Cliente
Id de requisito:	13
Nombre del requisito:	Comprar - Pago
Proceso realizado por el usuario:	Una vez tengamos todo lo que queremos en el carrito se procederá a realizar el pago.
Proceso realizado por el sistema:	El sistema reconoce que el cliente quiere pagar, calcula el total y presenta las opciones de pago disponibles.
Resultado de la interacción:	Se pasa a la ventana de selección de tipo de pago y una vez realizado se procede con el pago de este.

Tabla 3.14. Cliente - Comprar – Pago

3.2. Casos de uso

En este apartado mostraremos los casos de uso como diagramas utilizados para describir las acciones que realizarán los distintos grupos de usuarios: los Administradores, los Empleados y los Clientes. De esta forma mostraremos las posibles acciones o interacciones de los distintos grupos de usuarios con el sistema de una manera más grafica.

Casos de uso Administradores

Comenzaremos con los casos de uso principales del grupo de Administradores, tanto los genéricos cómo los más específicos. En primer lugar, podemos observar en la ilustración **3.1**, las interacciones que puede tener el administrador con el sistema una vez Inicie Sesión en el mismo, las cuales le darán acceso a los módulos de Contactos, Empleados y al resto de aplicaciones además de a los ajustes.

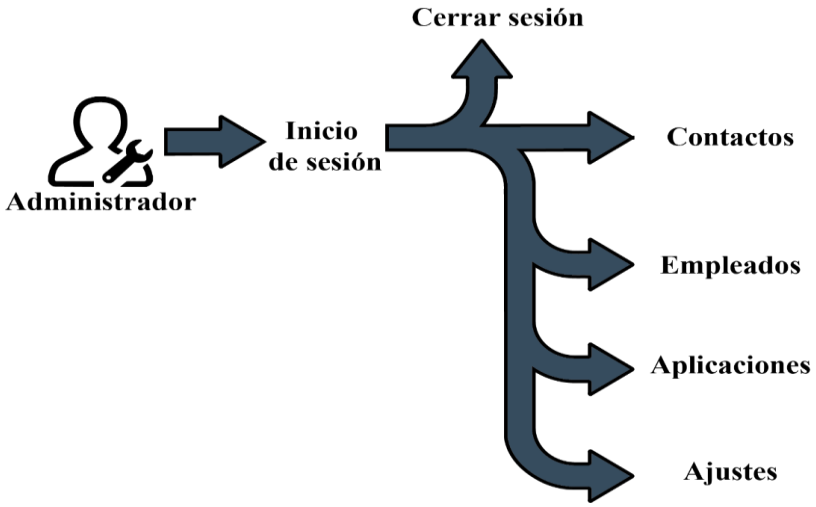


Ilustración 3. 1. Administrador - Uso general

En segundo lugar, pasaremos a los casos de uso más específicos de los Administradores, como podemos ver en la tabla **3.2**, una vez los administradores entren en la aplicación Contactos, podrán ver y editar la información de los clientes así como generar el horario de reparto. En la tabla **3.3** se muestra como accediendo a la aplicación de Empleados, se podrá general en nuevo documento que será un contrato de confidencialidad

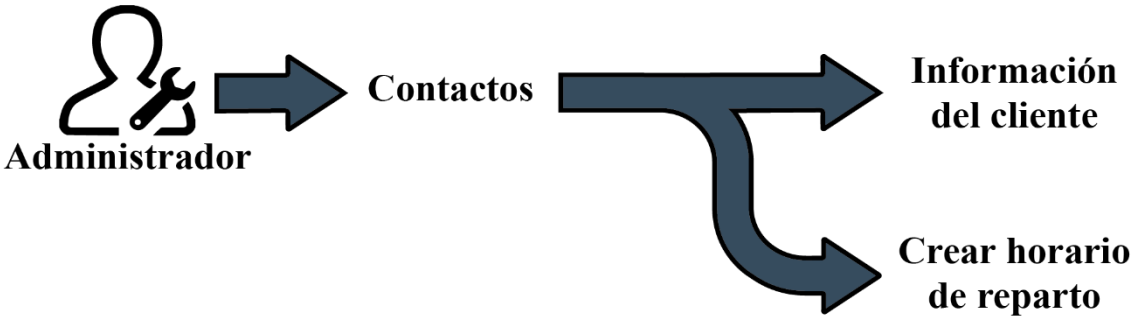


Ilustración 3. 2. Administrador - Gestión de clientes

Ilustración 3. 3. Administrador - Gestión de empleados

Casos de uso Clientes

En segundo lugar, mostraremos los casos de uso de los clientes. Comenzaremos por el caso de uso principal que será una vez el cliente entre en la web e inicie sesión. En este caso en cliente podrá interactuar con el sistema como se muestra en el caso de uso de la ilustración **3.4**.

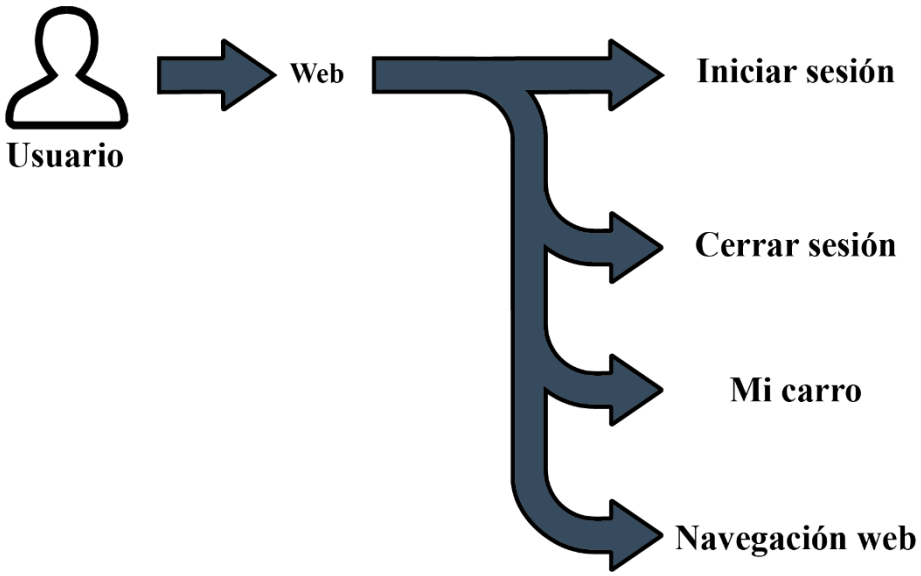


Ilustración 3. 4. Cliente - Uso general

Para acabar se mostrará el caso de uso de cuando el cliente tiene los artículos deseados en la cesta, lo que podrá hacer con ellos y como finalizar la transacción como se muestra en la ilustración **3.5**.

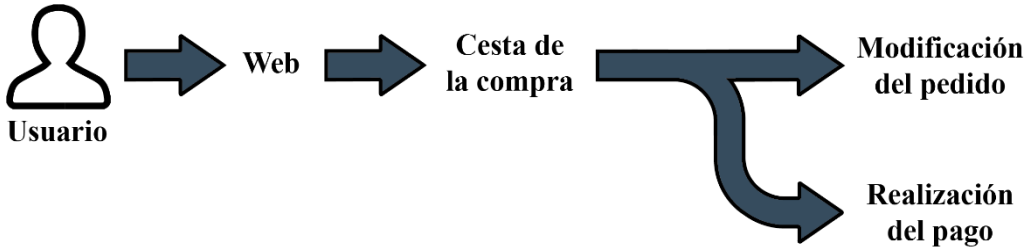


Ilustración 3. 5. Cliente - Proceso de compra

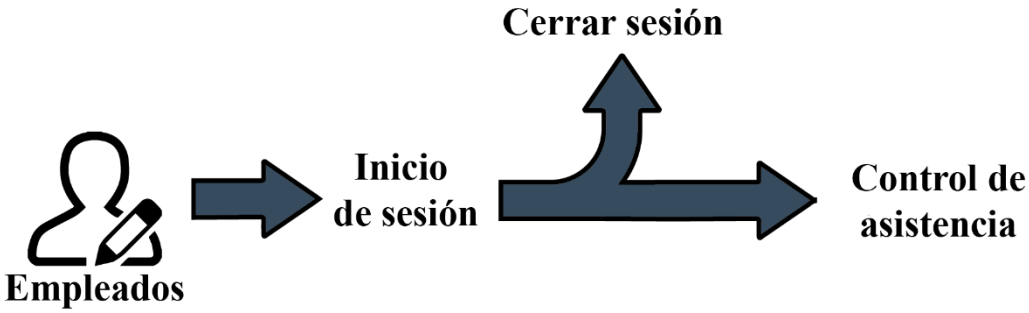


Ilustración 3. 6. Empleados - Uso general

4. Diseño de la solución

Una vez hemos analizado los requisitos del sistema a desarrollar y han quedado claras las necesidades que el mismo debe satisfacer podemos pasar a mostrar cómo va a ser el diseño de la solución. Esto se realizará indicando los módulos que conformarán la instalación y explicando su funcionalidad y mostrando una serie de prototipos que mostrarán cómo queremos que quede el despliegue una vez hayamos acabado con las modificaciones. Como parte de estas modificaciones consisten en la generación de documentos creados por mi, mostraremos también unos bocetos de cómo serán estos documentos antes y después de ser cumplimentados por el programa.

4.1. Módulos que componen el despliegue

Como se ha comentado con anterioridad en este apartado mostraremos y explicaremos los módulos que van a formar la instalación. Y de qué forma podrán satisfacer las necesidades de los requisitos presentados. Los módulos son los siguientes:

- **Asistencia.** Esta aplicación hace posible registrar la entrada y la salida de los trabajadores y así poder contrastar esa información con el resto de la información disponible. Esta aplicación nos servirá para poder controlar a los empleados y poder conocer o identificar cualquier error humano en el cambio de turno para poder así optimizar los tiempos durante este proceso.
- **Empleados.** Esta aplicación nos va a servir, en el contexto de este despliegue, para gestionar toda la información de los trabajadores de la empresa ya que nos permita centralizarla toda en un mismo lugar. Esta aplicación está relacionada con la mencionada anteriormente la de Asistencia, permitiéndonos así un control total sobre las asistencias y turnos de los empleados. Además, sobre este módulo realizaremos una de las ampliaciones que nos permitirá desde dentro de cada empleado crear un contrato de confidencialidad que se auto rellene con los datos del empleado seleccionado.
- **Ecommerce.** La tercera aplicación a mencionar se trata de Ecommerce, con esta aplicación podremos crear un comercio electrónico de manera sencilla. En este caso la utilizaremos para crear un sitio en donde los clientes (bares en nuestro caso) puedan realizar sus pedidos sin necesidad de tener que hacerlo de forma manual por parte de los gerentes de la empresa,

es decir, que no tengan que llamar día tras día para saber que cantidad de producto va a necesitar el cliente al día siguiente, reduciendo así la carga que se impone sobre los jefes del negocio.

- **Sitio web.** Esta aplicación va ligada completamente con la anterior, ya que es la que nos permite crear y modificar de manera muy sencilla nuestro sitio web desde donde podrán acceder los clientes para realizar sus pedidos. Con su configuración de arrastrar y soltar se ha facilitado en gran medida no solo la creación de la web, sino también las modificaciones puntuales o la ampliación y reducción de productos disponibles en un momento dado.
- **Facturación.** La siguiente aplicación nos ayudará con la gestión de las facturas, además de facilitar el manejo de los pagos y poder generar informes automatizados. Gracias a este módulo podremos facilitar en gran medida el proceso de facturación, podremos convertir presupuestos en facturas de la forma más sencilla posible y podremos de esta forma obtener una visión más detallada de las ventas de la empresa.
- **Contactos.** Por último, mencionaremos la aplicación de contactos, donde se podrá controlar y gestionar en un único lugar todos los clientes de la empresa de una manera rápida y simplificada, evitando así tener miles de datos sin registrar sobre todos los clientes de la empresa. También utilizaremos esta aplicación para facilitar el trabajo a los repartidores, ya que mediante la ampliación realizada se podrá generar un documento donde se muestren los clientes a los cuales se deben llevar los pedidos , desde su nombre y dirección hasta el horario de reparto además de un pequeño campo de texto en el que se podrán añadir anotaciones específicas para el repartidor, como por ejemplo especificaciones sobre el reparto o posibles medidas a tener en cuenta a la hora de entregar el pedido, para respetar así las medidas COVID de ser necesario.

4.2. Prototipos

El último apartado del diseño de la solución mostrará cuales han sido los bocetos de cómo se vería la interfaz de usuario referente a la ampliación de los módulos propuestos, es decir el de Empleados y el de Contactos, para poder dar una idea del problema a abarcar. En concreto se mostrarán las vistas de cómo quedará la vista de Contactos con el botón que generará el wizard en el cual introduciremos los datos para crear el documento de confidencialidad y el que se utilizará en la vista de Empleados para crear el documento con las direcciones y datos de reparto. También mostraremos los prototipos de los documentos creados que se rellenarán con los datos seleccionados y serán generados por Odoo ya cumplimentados listos para su firma. Hay que remarcar que estos bocetos son prototipos del diseño a implementar por lo que el diseño final podría llegar a variar.

Empezaremos mostrando como quedaran las vistas predefinidas de Odoo para las aplicaciones de Contactos y Empleados una vez añadamos los botones necesarios para la generación de los documentos. En contactos como podemos observar en la imagen **4.1** existirá el botón para generar el horario de reparto en el cual podremos añadir los distintos clientes que queremos figuren en el documento. En la aplicación de empleados encontraremos el botón generar documento de confidencialidad como podemos observar en la imagen **4.2**, donde al clicarlo se generará el acuerdo de confidencialidad.

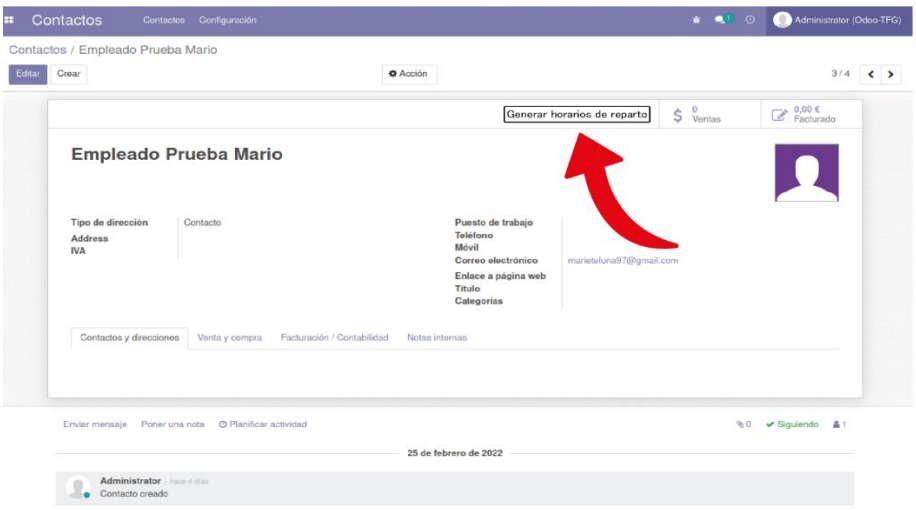


Ilustración 4. 1. Prototipo - Generar horarios de reparto

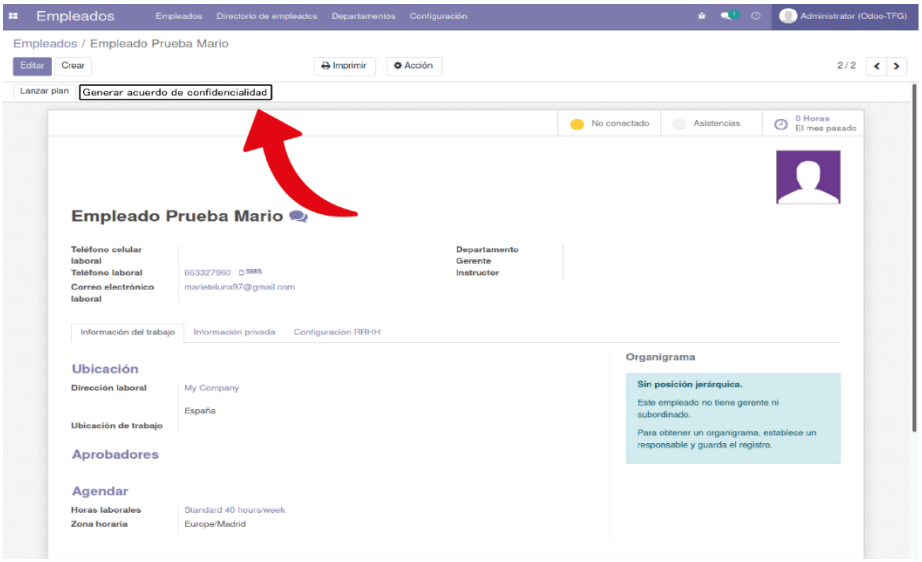


Ilustración 4. 2. Prototipo - Generar acuerdo de confidencialidad

En segundo lugar, mostraremos los bocetos de los prototipos creados con la aplicación Pencil, estos bocetos servirán para ver los wizards que aparecerán al darle a cada uno de los botones en los cuales seleccionaremos los contactos o empleados deseados y se generarán los documentos mencionados anteriormente. En la imagen **4.3** podemos observar el wizard que aparecerá al seleccionar el botón de la imagen **4.1** y que nos permitirá seleccionar todos los clientes que queremos que aparezcan en la hoja de reparto así como un pequeño cuadro de texto en el que introduciremos los detalles de la entrega. Por último, mencionar la imagen **4.4** en la que aparece un pequeño wizard en el que se seleccionaran los participantes en la firma del documento de confidencialidad, así como la fecha y el lugar de la firma. Este último wizard podría no llegar a aparecer en la versión definitiva ya que de ser posible se generará automáticamente con los datos que ya aparecen en la propia ficha del empleado.

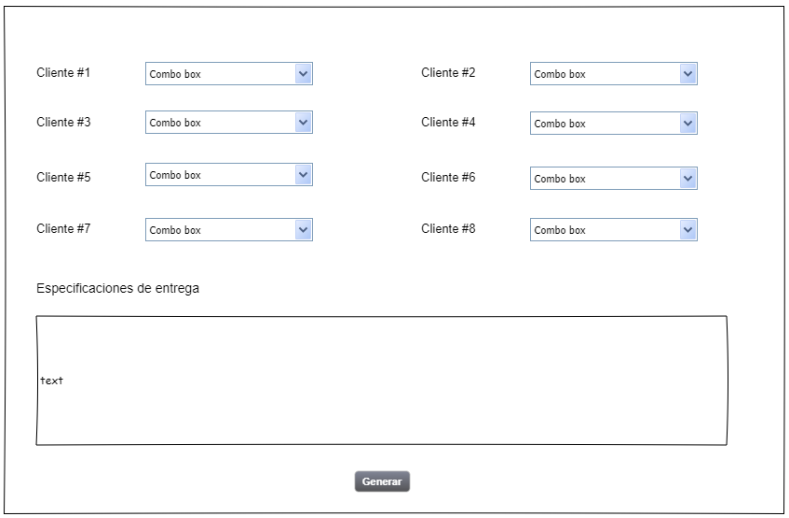
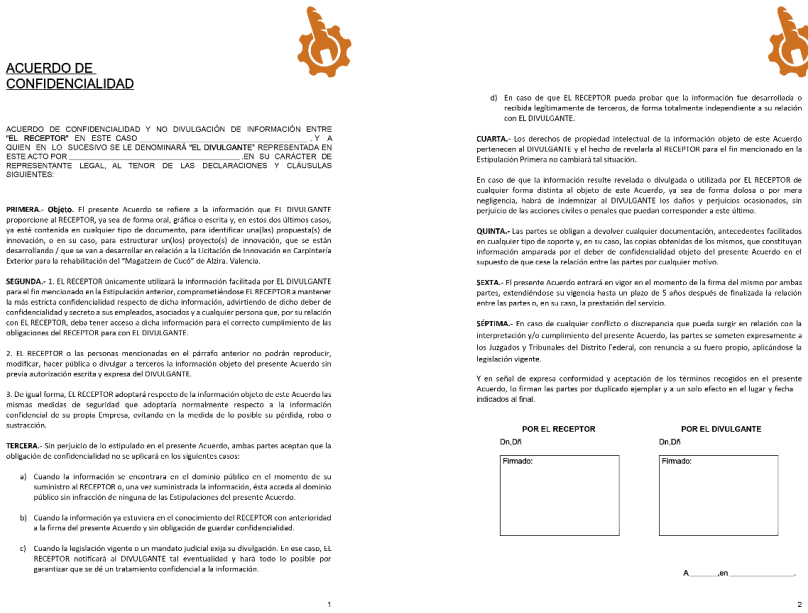


Ilustración 4. 3. Prototipo - Wizard generar hoja de reparto

Ilustración 4. 4. Prototipo - Wizard generar acuerdo de confidencialidad

A continuación, en las imágenes **4.5** y **4.6** podemos observar el diseño de los documentos que se han creado con el propósito de ser rellenados y generados por el sistema. En la imagen **4.5** podemos observar el documento que hace referencia al acuerdo de confidencialidad y en la imagen **4.6** podemos ver el documento que se utilizará para crear una hoja con los nombres y datos de reparto de los distintos clientes.



ACUERDO DE CONFIDENCIALIDAD

ACUERDO DE CONFIDENCIALIDAD Y NO DIVULGACIÓN DE INFORMACIÓN ENTRE "EL RECEPTOR" EN ESTE CASO _____ Y A QUIEN EN LO SUCESIVO SE LE DENOMINARÁ "EL DIVULGANTE" REPRESENTADA EN ESTE ACTO POR _____ .EN SU CARÁCTER DE REPRESENTANTE LEGAL, AL TENOR DE LAS DECLARACIONES Y CLÁUSULAS SIGUIENTES:

PRIMERA.- Objeto. El presente Acuerdo se refiere a la información que El DIVULGANTE proporcione al RECEPTOR, ya sea de forma oral, gráfica o escrita y, en estos dos últimos casos, ya esté contenida en cualquier tipo de documento, para identificar una(las) propuesta(s) de innovación, o en su caso, para estructurar un(los) proyecto(s) de innovación, que se están desarrollando / que se van a desarrollar en relación a la Licitación de Innovación en Carpintería Exterior para la rehabilitación del "Magatzem de Cucó" de Alzira. Valencia.

SEGUNDA.- 1. EL RECEPTOR únicamente utilizará la información facilitada por EL DIVULGANTE para el fin mencionado en la Estipulación anterior, comprometiéndose EL RECEPTOR a mantener la más estricta confidencialidad respecto de dicha información, advirtiendo de dicho deber de confidencialidad y secreto a sus empleados, asociados y a cualquier persona que, por su relación con EL RECEPTOR, deba tener acceso a dicha información para el correcto cumplimiento de las obligaciones del RECEPTOR para con EL DIVULGANTE.

2. EL RECEPTOR o las personas mencionadas en el párrafo anterior no podrán reproducir, modificar, hacer pública o divulgar a terceros la información objeto del presente Acuerdo sin previa autorización escrita y expresa del DIVULGANTE.

3. De igual forma, EL RECEPTOR adoptará respecto de la información objeto de este Acuerdo las mismas medidas de seguridad que adoptaría normalmente respecto a la información confidencial de su propia Empresa, evitando en la medida de lo posible su pérdida, robo o sustracción.

TERCERA.- Sin perjuicio de lo estipulado en el presente Acuerdo, ambas partes aceptan que la obligación de confidencialidad no se aplicará en los siguientes casos:

a) Cuando la información se encontrara en el dominio público en el momento de su suministro al RECEPTOR o, una vez suministrada la información, ésta acceda al dominio público sin infracción de ninguna de las Estipulaciones del presente Acuerdo.

b) Cuando la información ya estuviera en el conocimiento del RECEPTOR con anterioridad a la firma del presente Acuerdo y sin obligación de guardar confidencialidad.

c) Cuando la legislación vigente o un mandato judicial exija su divulgación. En ese caso, EL RECEPTOR notificará al DIVULGANTE tal eventualidad y hará todo lo posible por garantizar que se dé un tratamiento confidencial a la información.

1

d) En caso de que EL RECEPTOR pueda probar que la información fue desarrollada o recibida legítimamente de terceros, de forma totalmente independiente a su relación con EL DIVULGANTE.

CUARTA.- Los derechos de propiedad intelectual de la información objeto de este Acuerdo pertenecen al DIVULGANTE y el hecho de revelarla al RECEPTOR para el fin mencionado en la Estipulación Primera no cambiará tal situación.

En caso de que la información resulte revelada o divulgada o utilizada por EL RECEPTOR de cualquier forma distinta al objeto de este Acuerdo, ya sea de forma dolosa o por mera negligencia, habrá de indemnizar al DIVULGANTE los daños y perjuicios ocasionados, sin perjuicio de las acciones civiles o penales que puedan corresponder a este último.

QUINTA.- Las partes se obligan a devolver cualquier documentación, antecedentes facilitados en cualquier tipo de soporte y, en su caso, las copias obtenidas de los mismos, que constituyan información amparada por el deber de confidencialidad objeto del presente Acuerdo en el supuesto de que cese la relación entre las partes por cualquier motivo.

SEXTA.- El presente Acuerdo entrará en vigor en el momento de la firma del mismo por ambas partes, extendiéndose su vigencia hasta un plazo de 5 años después de finalizada la relación entre las partes o, en su caso, la prestación del servicio.

SÉPTIMA.- En caso de cualquier conflicto o discrepancia que pueda surgir en relación con la interpretación y/o cumplimiento del presente Acuerdo, las partes se someten expresamente a los Juzgados y Tribunales del Distrito Federal, con renuncia a su fuero propio, aplicándose la legislación vigente.

Y en señal de expresa conformidad y aceptación de los términos recogidos en el presente Acuerdo, lo firman las partes por duplicado ejemplar y a un solo efecto en el lugar y fecha indicados al final.

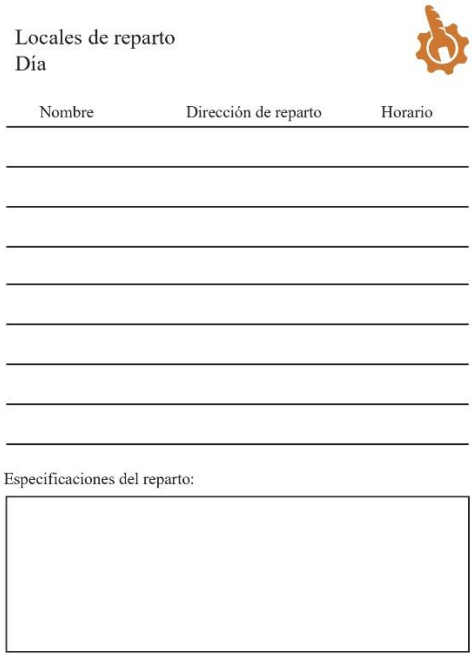
POR EL RECEPTOR
Dn.Dñ
Firmado:

POR EL DIVULGANTE
Dn.Dñ
Firmado:

A_____,en_____.

2

Ilustración 4. 5. Prototipo - Acuerdo de confidencialidad



Locales de reparto
Día

Nombre	Dirección de reparto	Horario

Especificaciones del reparto:

Ilustración 4. 6. Prototipo - Hoja de reparto

5. Desarrollo de la solución

En este capítulo se explicará el desarrollo del proyecto de una forma más profunda y exhaustiva, incluyendo fragmentos de código de los modelos, así como de las vistas, luego se mostrará la composición del directorio que contiene todo el programa y también como se ha utilizado la herencia para poder realizar las ampliaciones de las aplicaciones. Por último, se explicará como hemos realizado las modificaciones de Odoo para que adquiera las funcionalidades deseadas como por ejemplo el diseño de la web o la configuración de acceso de los usuarios a las distintas aplicaciones.

5.1. Directorio

El directorio del proyecto está compuesto por varias carpetas y archivos que conforman todo el módulo que aporta las ampliaciones de funcionalidad como podemos ver en la imagen **5.1**.

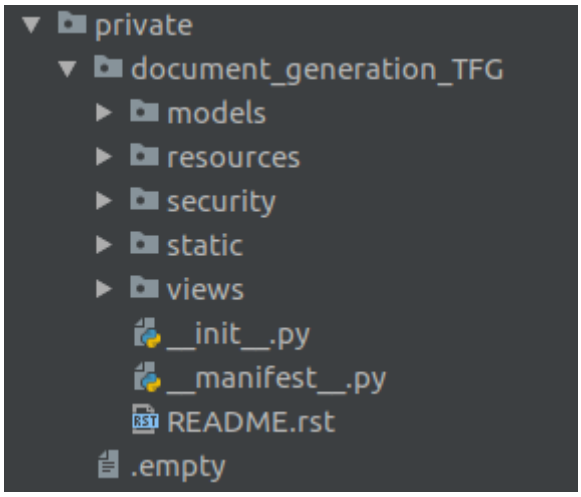


Ilustración 5. 1. Directorio del proyecto

Como podemos ver hay numerosos directorios que conforman la totalidad del módulo que ha servido para ampliar la funcionalidad del módulo de Empleados y el de Contactos. Ahora se explicarán los siguientes directorios y documentos de una manera sencilla y a posteriori se realizará una definición detallada de los que necesiten un estudio más profundo de su funcionalidad. Estos directorios y documentos son:

- **models.** En este directorio están contenidos los modelos u objetos que componen nuestra aplicación de Odoo.
- **resources.** En este directorio están contenidos los documentos PDF que hemos creado para las ampliaciones. En nuestro caso el acuerdo de confidencialidad y la hoja de repartos como se muestran en la imagen 5.2.

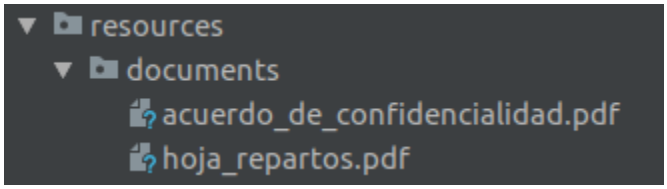


Ilustración 5. 2. Carpeta resources

- **security.** Este directorio contiene la configuración de seguridad de nuestro proyecto. En este caso la seguridad hace referencia a los permisos de acceso a la funcionalidad de las ampliaciones.
- **static.** Este directorio sirve para guardar los datos estáticos, en nuestro caso el icono de la aplicación como se muestra en la imagen 5.3. Pero también suele utilizarse para almacenar los archivos JavaScript o CSS que sirven para darle un estilo más personalizado a las aplicaciones.

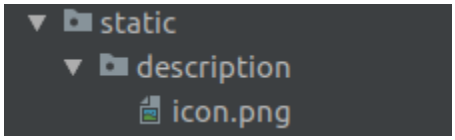


Ilustración 5. 3. Carpeta static

- **views.** En este directorio se almacenan los archivos con formato XML, estos son las vistas que van a ser ampliadas por nuestro módulo y una nueva.
- **_init_.py.** En este archivo se encuentra una única línea de código que sirve para indicar que el sistema debe cargar los módulos que se encuentren en la carpeta de models.
- **_manifest_.py.** Este archivo sirve para indicar los metadatos del módulo desarrollado en la lista en la que se muestran todas las aplicaciones disponibles de Odoo. La estructura de este se puede ver a continuación, sobre esta estructura deberíamos mencionar que en el

apartado data se incluirán todas las vistas y archivos de seguridad, para que se carguen una vez iniciada la aplicación. También es importante mencionar los apartados depends, en el cual se añaden las dependencias y los módulos de los que dependerá el nuestro, images, el cual carga el icono de la aplicación, e instalable, que nos indica si la aplicación será instalable.

Ilustración 5. 4. Contenido del manifest

- **README.rst.** Y por último este archivo contiene información de la aplicación para dar información sobre él, en este caso lo he utilizado únicamente para indicar el nombre del módulo, pero se podría añadir perfectamente una descripción del mismo.

5.2. Herencia

Antes de continuar explicando el resto de los directorios importantes del módulo, se explicará cómo funciona la herencia en Odoo ya que ha sido un pilar fundamental para la ampliación de los módulos y el desarrollo de este proyecto.

En el mundo de la tecnología y el software la herencia se utiliza sobre todo en la programación orientado a objetos, es el segundo mecanismo más utilizado en el desarrollo de software después de la agregación. Gracias a este mecanismo se pueden crear nuevas clases partiendo de una única clase ya existente o de una jerarquía de clases, de este modo se evita tener que rediseñar, verificar o modificar la parte que ya se encuentra implementada. Este mecanismo nos facilita la creación de objetos a partir de otros ya existentes, de esta forma una subclase siempre obtendrá las variables y los métodos de su superclase.

Ahora que ya se ha explicado que es la herencia, podemos pasar a explicar cómo funciona este mecanismo en Odoo(**13**).

La herencia en Odoo puede ser utilizada tanto en los modelos y en las vistas, y en cada uno nos aporta una serie de ventajas distintas pero manteniendo su funcionalidad principal mencionada en los párrafos anteriores. Se comenzará por explicar la herencia en los modelos, en este caso son los contenidos en la carpeta models explicada con anterioridad.

Odoo proporciona dos mecanismos de herencia que se pueden utilizar para extender un modelo ya existente. Con el primero de estos mecanismos de herencia se puede utilizar un módulo para modificar el comportamiento de un modelo perteneciente a otro módulo, estas son las cosas que podemos hacer con este mecanismo:

- Agregar restricciones a un modelo.
- Agregar métodos a un modelo.
- Agregar campos a un modelo.
- Anular los métodos existentes en un modelo.
- Anular la definición de campos de un modelo.

El segundo de estos mecanismos es la herencia de delegación mediante el cual se puede vincular cada uno de los registros de un modelo a los registros en un modelo principal, proporcionando así acceso transparente a los campos del registro del modelo principal.

Se pueden ver ambos tipos de herencia en la siguiente imagen obtenida directamente de la web de Odoo (13).

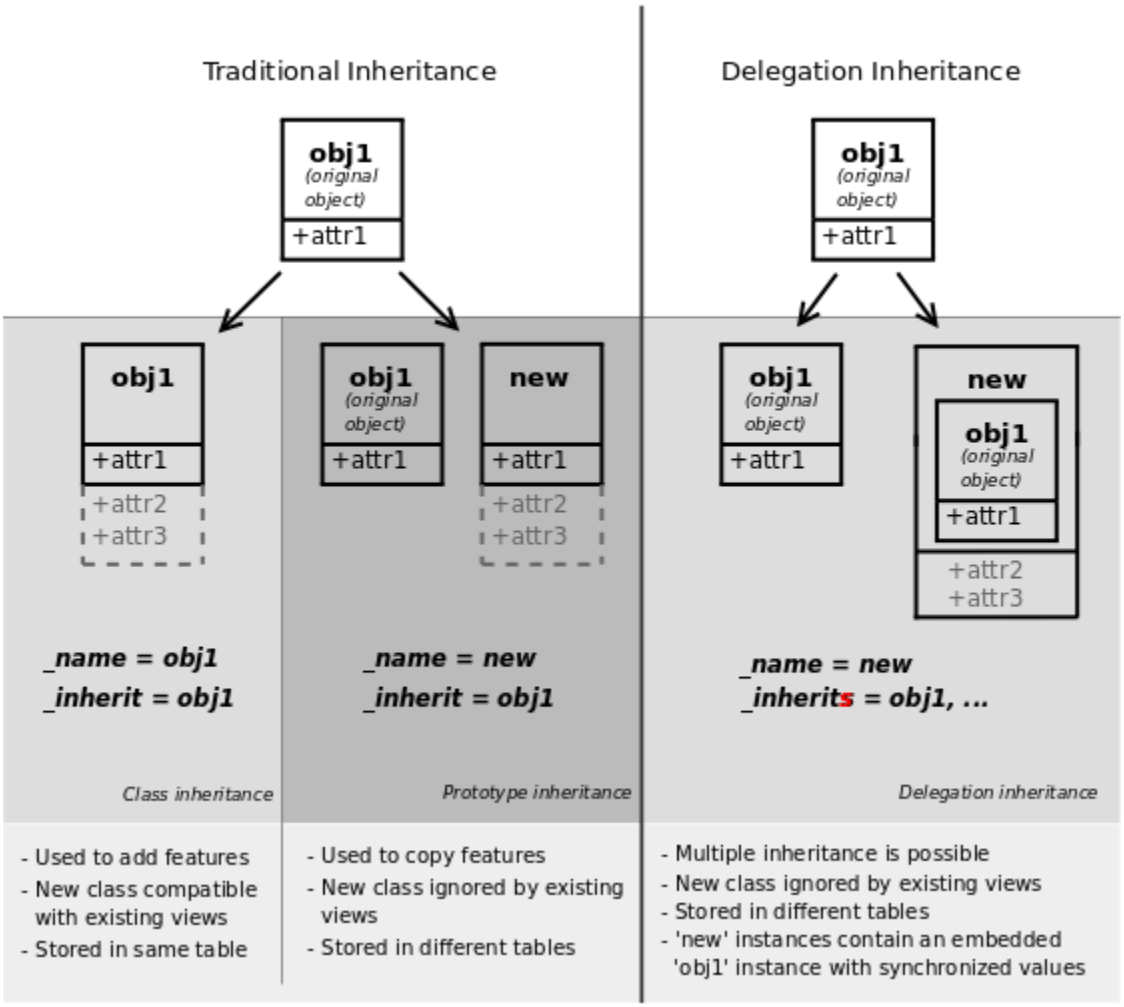


Ilustración 5. 5. Herencia de Odoo (13)

A continuación se explicará cómo funciona la herencia en las vistas. Gracias a la herencia se pueden crear vistas cuyas modificaciones se podrán aplicar sobre las vistas raíz sobrescribiéndolas y de esta manera agregar o eliminar contenido de las vistas del padre.

Las vistas que utilizan la herencia hacen referencia a su vista padre mediante el campo *inherit_id* y la estructura de su cuerpo puede contener tantos *xpath* como queramos, estos campos sirven para identificar un elemento de la vista padre y acto seguido se puede o modificarlos o añadir nuevas funcionalidades y mediante el atributo *position* se pueden posicionar nuevos campos delante, detrás, dentro de los elementos o incluso sustituirlos.

```
<record id="delivery_timetable_form_view_tfg" model="ir.ui.view">
    <field name="name">res.partner.custom.inherit.form.delivery</field>
    <field name="model">res.partner</field>
    <field name="inherit_id" ref="base.view_partner_form" />
    <field name="arch" type="xml">
        <xpath expr="//field[@name='vat']" position="after">
            <field name="contact_type" />
            <field name="time" widget="float_time"  attrs="{'invisible': [
        </xpath>
```

Ilustración 5. 6. Herencia en vistas

Una vez explicado que es la herencia y cómo funciona en Odoo se ha de explicar la importancia de esta en el desarrollo de este proyecto. Aparte de las modificaciones que se han realizado sobre Odoo que se comentarán más adelante, el proyecto se basado mayormente en añadir de nueva funcionalidad a módulos ya existentes. De esta forma mediante la herencia se han utilizado las vistas ya existentes de Contactos y Empleados y se han ampliado para mostrar los botones que sirven para activar las nuevas funcionalidades de las ampliaciones desarrolladas

5.3. Models

En Odoo los modelos son los objetos que son implementados en nuestra aplicación mediante clases Python. Nuestro directorio de models está compuesto por cuatro archivos, el primero de ellos el **__init__.py** sirve para inicializar los modelos que se han desarrollado, su estructura sería como se muestra a continuación.

```
from . import res_partner
from . import delivery_timetable
from . import hr_employee
```

Ilustración 5. 7. Contenido del init

En primer lugar se comentará el modelo **res_partner**, este es un modelo sencillo, en el cual simplemente se ha creado unos cuantos campos que servirán para ser utilizados por el modelo **delivery_timetable** o para dar especial funcionalidad al mismo. El código es el siguiente:

```
class ResPartnerContract(models.Model):
    _inherit = "res.partner"

    delivery_ids = fields.One2many(
        comodel_name='delivery.timetable',
        inverse_name='partner_id',
        string='Documento de reparto',
    )
    contact_type = fields.Selection(
        selection=[
            ('client', 'Cliente'),
            ('Worker', 'Empleado'),
            ('Other', 'Otro'),
        ],
        string='Tipo de contacto',
        help='Selecciona el tipo de contacto',
    )
    time = fields.Float(
        string='Time',
    )
```

Ilustración 5. 8. Model res_partner

Los campos que se pueden ver en la imagen superior tienen funcionalidades diferentes, esas funcionalidades son las siguientes:

- **_inherit.** Este campo indica de que modelo heredara nuestra clase, en este caso se trata de res.partner que haría referencia al modelo de Contactos.
- **delivery_ids.** Gracias a este campo se puede obtener los datos del modelo de delivery_timetable y así crear en la vista de Contactos un pequeño menú para mostrar todas las hojas de reparto creadas.
- **contact_type.** Este campo es una creación propia y gracias a él podremos otorgar un valor especial a cada contacto, para que solo los Contactos con la categoría de 'Otros' puedan visualizar y acceder al registro de las hojas de reparto para poder revisarlos o crear alguno nuevo.
- **time.** Este es un campo sencillo que permitirá asignar un horario a los clientes para saber a qué hora desean que sus comandas sean entregadas diariamente.

A continuación se encuentra el modelo **delivery_timetable**, este modelo sirve para crear las funciones y los atributos que servirán para crear la hoja de reparto como se muestra a continuación en la siguiente figura. En primer lugar, se encuentran los siguientes atributos:

- **_name.** Este campo indica el nombre del modelo para su mención a posterioridad, como por ejemplo para usarlo para crear las vistas como se verá en el siguiente apartado.
- **_description**. Este no es un campo necesario pero sirve para dar una pequeña descripción de lo que hace este modelo.
- **partner_id.** Este campo sirve para asociar este modelo con el de res_partner ya que la vista principal del modelo de delivery se verá como una vista en forma lista dentro de la vista de partner y al seleccionar en ella se abrirá un wizard para rellenar los datos del documento.
- **name_id.** Sirve para otorgarle a la hoja de reparto generada un nombre identificativo como por ejemplo "Repartos fallas".
- **document_date.** Servirá para otorgarle una fecha a la hoja de reparto, esto servirá tanto para crear los repartos diarios como para identificarlos por la fecha de su creación.
- **specs.** Este campo sirve para almacenar y luego plasmar en la hoja de reparto cualquier comentario especial, ya sea de la ruta o a la hora de la entrega.

Para los siguientes campos de partner hay que indicar que están numerados debido a que existen ocho de cada, para la creación de cada uno de los clientes junto con su dirección y el horario de entrega. Estos son:

- **partner_1_id.** Indica el nombre del cliente.
- **partner_1_street, city, zip.** Juntos forman la dirección completa del cliente.
- **partne_1_time.** Indica la hora de reparto para ese cliente en concreto.

```
class DeliveryTimetable(models.Model):
    _name = 'delivery.timetable'
    _description = 'Delivery Timetable'

    partner_id = fields.Many2one(
        comodel_name='res.partner',
        string='Nombre del cliente',
    )
    name_id = fields.Char(
        string='Nombre identificativo del documento',
    )
    document_date = fields.Date(
        string='Fecha',
    )
    specs = fields.Text(
        string='Especificaciones de reparto'
    )
    partner_1_id = fields.Many2one(
        comodel_name='res.partner',
        string='Nombre del cliente',
    )
    partner_1_street = fields.Char(
        string='Calle del cliente',
    )
    partner_1_city = fields.Char(
        string='Ciudad del cliente',
    )
    partner_1_zip = fields.Char(
        string='Zip del cliente',
    )
    partner_1_time = fields.Float(
        string='Time',
    )
```

Ilustración 5. 9. Model delivery_timetable

Este modelo no hereda de ningún otro modelo, es uno original cuya funcionalidad ha sido mencionada anteriormente. Por tanto para que realizase lo necesario para crear el documento se crearon una serie de funciones. La primera se trata de un onchange, el cual es una función que cuando se asigne un partner mediante **partner_X_id**, asignará al resto de los atributos partner con ese número X los valores correspondientes según el cliente. Este sería el código:

```
@api.onchange('partner_1_id')
def _onchange_partner_1_id(self):
    for record in self:
        record.partner_1_street = record.partner_1_id.street
        record.partner_1_city = record.partner_1_id.city
        record.partner_1_zip = record.partner_1_id.zip
        record.partner_1_time = record.partner_1_id.time
```

Ilustración 5. 10. Función compute

El siguiente paso es utilizar el PDF editable y mediante la función **generate_delivery_timetable** ir rellenando el documento con los atributos creados en el modelo. En primer lugar tendremos las variables que se han utilizado para combinar los campos city, street y zip para crear la dirección completa, así como generar los atributos que almacenaran el PDF antes y después de ser procesado.

```
def generate_delivery_timetable(self):
    """
    Generamos la hoja de reparto.
    """
    partner_1_full_address = '%s, %s, %s' % (self.partner_1_street or"",
                                             self.partner_1_city or "",
                                             self.partner_1_zip or "")
    partner_2_full_address = '%s, %s, %s' % (self.partner_2_street or "",
                                             self.partner_2_city or "",
                                             self.partner_2_zip or "")
    partner_3_full_address = '%s, %s, %s' % (self.partner_3_street or "",
                                             self.partner_3_city or "",
                                             self.partner_3_zip or "")
    partner_4_full_address = '%s, %s, %s' % (self.partner_4_street or "",
                                             self.partner_4_city or "",
                                             self.partner_4_zip or "")
    partner_5_full_address = '%s, %s, %s' % (self.partner_5_street or "",
                                             self.partner_5_city or "",
                                             self.partner_5_zip or "")
    partner_6_full_address = '%s, %s, %s' % (self.partner_6_street or "",
                                             self.partner_6_city or "",
                                             self.partner_6_zip or "")
    partner_7_full_address = '%s, %s, %s' % (self.partner_7_street or "",
                                             self.partner_7_city or "",
                                             self.partner_7_zip or "")
    partner_8_full_address = '%s, %s, %s' % (self.partner_8_street or "",
                                             self.partner_8_city or "",
                                             self.partner_8_zip or "")
    pdf_delivery_timetable = (
        "/opt/odoo/custom/src/private/document_generation_TFG"
        "/resources/documents/hoja_repartos.pdf")
    pdf_delivery_timetable_generated = "/tmp/hoja_repartos.pdf"
    values = dict()
```

Ilustración 5. 11. Función generate_delivery_timetable parte 1

Una vez generados estos valores se puede pasar a rellenar el PDF como se puede observar en la figura **5.12**:

```
    # Valores del documento.
    values["date"] = self.document_date or ""
    values["cliente 1"] = self.partner_1_id.name or ""
    values["cliente 2"] = self.partner_2_id.name or ""
    values["cliente 3"] = self.partner_3_id.name or ""
    values["cliente 4"] = self.partner_4_id.name or ""
    values["cliente 5"] = self.partner_5_id.name or ""
    values["cliente 6"] = self.partner_6_id.name or ""
    values["cliente 7"] = self.partner_7_id.name or ""
    values["cliente 8"] = self.partner_8_id.name or ""
    values["direccion 1"] = partner_1_full_address or ""
    values["direccion 2"] = partner_2_full_address or ""
    values["direccion 3"] = partner_3_full_address or ""
    values["direccion 4"] = partner_4_full_address or ""
    values["direccion 5"] = partner_5_full_address or ""
    values["direccion 6"] = partner_6_full_address or ""
    values["direccion 7"] = partner_7_full_address or ""
    values["direccion 8"] = partner_8_full_address or ""
    values["horario 1"] = self.partner_1_id.time or ""
    values["horario 2"] = self.partner_2_id.time or ""
    values["horario 3"] = self.partner_3_id.time or ""
    values["horario 4"] = self.partner_4_id.time or ""
    values["horario 5"] = self.partner_5_id.time or ""
    values["horario 6"] = self.partner_6_id.time or ""
    values["horario 7"] = self.partner_7_id.time or ""
    values["horario 8"] = self.partner_8_id.time or ""
    values["specs"] = self.specs or ""
    values["nombre documento"] = self.name_id or ""
    # GENERAMOS LA HOJA DE REPARTO
    result = pypdftk.fill_form(pdf_delivery_timetable, values,
                               pdf_delivery_timetable_generated)
    if not result:
        raise UserError(_(
            'Error generando el documento de reparto'))
    return result
```

Ilustración 5. 12. Función generate_delivery_timetable parte 2

Y por último se encuentra el código de la función **generate_documents** que se ejecutara al darle al botón para generar la tabla, es decir el que generará el documento ya rellenado mediante la función mostrada anteriormente.

```
def generate_documents(self):
    """
    Generamos y adjuntamos la hoja de reparto.
    """
    self.ensure_one()
    document_name = self.generate_delivery_timetable()
    # creamos el adjunto
    doc = open(document_name, 'rb').read()
    doc_base64 = base64.encodebytes(doc)
    values_attach = {
        'mimetype': 'application/pdf',
        'name': 'Reparto-%s-%s.pdf' % (self.name_id,
                                       self.document_date),
        'res_model': self.partner_id._name,
        'datas': doc_base64,
        'res_id': self.partner_id.id,
    }
    self.env['ir.attachment'].create(values_attach)
```

Ilustración 5. 13. Función generate_documents

Y por último se encuentra el modelo de **hr_employee**, debido a que este modelo utiliza muchos de los campos ya provenientes del módulo de Empleados, su desarrollo ha sido más sencillo teniendo que crear únicamente un atributo. Su estructura sería la siguiente:

```
class HrEmployeeAgreement(models.Model):
    _inherit = 'hr.employee'

    agreement_date = fields.Date(
        string='Agreement Date',
        default=datetime.today()
    )
```

Ilustración 5. 14. Model hr_employee

- **_inherit.** Este campo lo que hace es indicarnos de que modelo heredará nuestra clase, es este caso hr_employee.
- **agreement_date.** Mediante este campo le asignaremos al contrato una fecha de creación dentro del mismo, el cual será por defecto la fecha actual a la hora de generar el contrato.

Las funciones de este modelo son muy similares a las del modelo anterior, en primer lugar se rellenará el PDF con la función **generate_confidentiality_agreement** mostrada en la imagen X y la función **generate_agreement**, imagen Y, servirá para que una vez clicado el botón se genere el acuerdo de confidencialidad. La diferencia entre estos modelos es que en el caso del acuerdo de confidencialidad no deberá rellenarse ningún campo ya que se utilizan los propios o heredados del modelo de Empleados.

```
def generate_confidentiality_agreement(self):
    """
    Generamos el Acuerdo de Confidencialidad.
    """
    pdf_confidentiality_agreement = (
        "/opt/odoo/custom/src/private/document_generation_TFG"
        "/resources/documents/acuerdo de confidencialidad.pdf")
    pdf_confidentiality_agreement_generated = \
        "/tmp/acuerdo_de_confidencialidad.pdf"
    values = dict()
    values["receptor"] = self.name or ""
    values["divulgante"] = self.parent_id.name or ""
    values["receptor_2"] = self.name or ""
    values["divulgante_2"] = self.parent_id.name or ""
    values["fecha"] = self.agreement_date or ""
    values["localizacion"] = self.address_id.city or ""
    # GENERAMOS EL ACUERDO
    result = pypdftk.fill_form(pdf_confidentiality_agreement, values,
                               pdf_confidentiality_agreement_generated)
    if not result:
        raise UserError(_(
            'Error generando el acuerdo de confidencialidad'))
    return result
```

Ilustración 5. 15. Función generate_confidentiality_agreement

```python
def generate_agreement(self):
    """
    Generamos el Acuerdo de Confidencialidad.
    """
    self.ensure_one()
    document_name = self.generate_confidentiality_agreement()
    # creamos el adjunto
    doc = open(document_name, 'rb').read()
    doc_base64 = base64.encodebytes(doc)
    values_attach = {
        'mimetype': 'application/pdf',
        'name': 'AcuerdoConfidencialidad-%s-%s.pdf' %
                (self.name, self.agreement_date),
        'res_model': self._name,
        'datas': doc_base64,
        'res_id': self.id,
    }
    self.env['ir.attachment'].create(values_attach)
```

Ilustración 5. 16. Función generate_agreement

5.4. Views

En este aparado se explicarán las vistas que se pueden encontrar en el directorio views. Todas estas vistas están en formato XML, las cuales serán convertidas a HTML por el sistema Odoo. Hay numerosos tipos de vista que se pueden utilizar en Odoo, form, tree, kanban, calendar, pero me centraré en las utilizadas en el proyecto que son la vista tree (lista) y la form (formulario).

Siguiendo el mismo orden que en el apartado anterior el primer archivo XML que se va a comentar es el de **res_partner.xml**. Este archivo tiene varias cosas importantes que deben ser mencionadas, en primer lugar se puede observar cómo mediante el campo inherit_id heredamos de la vista de partner para ampliarla. A continuación se observa un xpath que servirá para identificar un componen de la vista base e introducir nuestros campos detrás del mismo, estos son el campo de tipo de contacto y el de tiempo de reparto, el cual como se puede ver en la imagen **5.17** solo será visible si el contacto se trata de un cliente. Para finalizar hay que indicar que el campo notebook servirá para poder añadir la lista de hojas de reparto en una de las pestañas que se mostrarán al final de la vista de contactos y esta tendrá también un atributo para hacerla invisible pero en este caso solo se mostrará cuando el contacto no sea ni un cliente ni un empleado.

```xml
<odoo>
    <record id="delivery_timetable_form_view_tfg" model="ir.ui.view">
        <field name="name">res.partner.custom.inherit.form.delivery</field>
        <field name="model">res.partner</field>
        <field name="inherit_id" ref="base.view_partner_form" />
        <field name="arch" type="xml">
             <xpath expr="//field[@name='vat']" position="after">
                 <field name="contact_type" />
                 <field name="time" widget="float_time"
                        attrs="{'invisible': [('contact_type','!=','client')]}"/>
            </xpath>
            <notebook position="inside">
                <page string="Documentos de reparto"
                      attrs="{'invisible': [('contact_type','!=','Other')]}">
                    <field name="delivery_ids" nolabel="1"/>
                </page>
            </notebook>
        </field>
    </record>
</odoo>
```

Ilustración 5. 17. Vista partner

En segundo lugar encontramos las vistas del modelo delivery_timetable, este modelo tiene dos vistas, una tree, como se puede observar en la imagen **5.18**, y otra form, imagen **5.19**, ambas contenidas en el archivo **delivery_views.xml**. La vista tree de este modelo servirá para indicar que campos se quiere que se muestren en el listado de hojas de reparto dentro de la aplicación de contactos, de esta forma tendrá un campo identificativo con el nombre del documento, otro con su fecha de creación y por último contendrá el botón que generará la hoja de reparto correspondiente. Su vista form sin embargo tendrá una funcionalidad diferente, esta vista servirá para que al crear el registro de la nueva hoja se abra un pequeño formulario donde se podrá asignarle los ocho clientes que se incluirán en la hoja, junto con el nombre identificativo del documento, su fecha y un pequeño campo para dar indicaciones adicionales al repartidor para la entrega de los pedidos. Debido a la gran extensión del código de la vista form y que se repetirá la misma estructura para todos los clientes solo se mostrará el principio

```xml
<!-- Tree View -->
<record model="ir.ui.view" id="view_delivery_timetable_tree">
    <field name="name">delivery.timetable.tree</field>
    <field name="model">delivery.timetable</field>
    <field name="arch" type="xml">
        <tree>
            <field name="name_id"/>
            <field name="document_date"/>
            <button
                name="generate_documents"
                type="object"
                string="Genar hoja reparto"
            />
        </tree>
    </field>
</record>
```

Ilustración 5. 18. Vista delivery parte 1

```
<!--     Form View-->
    <record model="ir.ui.view" id="view_delivery_timetable_form">
        <field name="name">delivery.timetable.form</field>
        <field name="model">delivery.timetable</field>
        <field name="arch" type="xml">
            <form>
                <sheet>
                    <group>
                        <group string="Datos del documento">
                            <field name="name_id"/>
                            <field name="document_date"/>
                            <field name="specs"/>
                        </group>
                        <group>
                        </group>
                    </group>
                    <group>
                        <group string="Cliente 1">
                            <field name="partner_1_id" options="{'no_open':
                            True, 'no_create': True}"/>
                            <field name="partner_1_street"/>
                            <field name="partner_1_city"/>
                            <field name="partner_1_zip"/>
```

Ilustración 5. 19. Vista delivery parte 2

Por último hay que mencionar el archivo XML que hace referencia a la ampliación de la aplicación de los Empleados, el archivo **hr_employee.xml**. Este es el más simple de los tres, ya que como todos los campos necesarios ya se encontraban en el modelo base, lo único que se ha realizado ha sido agregar un botón en la parte superior de la vista que al ser clicado desencadenará la creación del acuerdo de confidencialidad. A continuación se puede observar su estructura en la imagen **5.20**.

```
<odoo>
    <record id="hr_employee_form_view_tfg" model="ir.ui.view">
        <field name="name">hr.employee.custom.inherit.form</field>
        <field name="model">hr.employee</field>
        <field name="inherit_id" ref="hr.view_employee_form" />
        <field name="arch" type="xml">
            <header>
                <button
                    name="generate_agreement"
                    type="object"
                    string="Genar Acuerdo de Confidencialidad"
                />
            </header>
        </field>
    </record>
</odoo>
```

Ilustración 5. 20. Vista employee

5.5. Security

Nuestro apartado de security únicamente contiene un archivo, ir.model.access.csv. Esto es debido a que como se utilizan los grupos ya predefinidos por Odoo, en nuestro caso el administrador y los empleados, no es necesario crear un grupo específico. Debido a que los empleados solo podrán acceder a la asistencia, los únicos que podrán acceder a la creación de las hojas de reparto son los administradores, por lo tanto un acceso de permisos general serviría para que estos puedan acceder, de todas formas se le asignará solo permiso de acceso a los administradores para evitar futuros errores o accesos indeseados.

```
id,name,model_id:id,group_id:id,perm_read,perm_write,perm_create,perm_unlink
access_delivery_timetable,delivery_timetable,model_delivery_timetable,base.group_erp_manager,1,1,1,1
```

Ilustración 5. 21. Permisos

5.6. Configuraciones en Odoo

A parte de las configuraciones realizadas por código se deberán realizar una serie de modificaciones en Odoo para que adopte el funcionamiento que se desea. La primera de las modificaciones será configurar el servidor de correo saliente desde **Ajustes** > **Técnico** > **Servidores de correo saliente**, esto ha sido realizado como podemos observar en la siguiente imagen, una vez realizado el botón de probar conexión debería devolver un mensaje diciendo que la conexión ha sido exitosa.

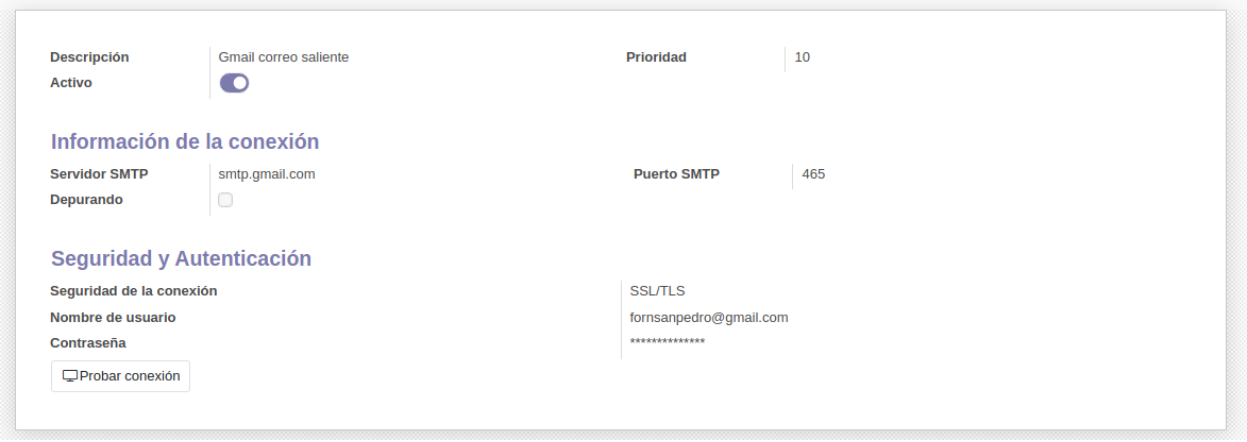


Ilustración 5. 22. Configuración correo saliente

En segundo lugar se configurarán a los usuarios creados para que cuando accedan los redireccione directamente a la página de asistencias, esto se hace desde **Ajustes > Usuarios y compañías > Usuarios** y ahí se accedería al usuario deseado. Una vez en el usuario, se accedería a la pestaña de **Preferencias** y una vez dentro se configuraría el apartado **Personalización de menús** como se muestra a continuación.

Ilustración 5. 23. Primer acceso

En tercer lugar se han realizado unas modificaciones para configurar los módulos a los que tendrán acceso los empleados. Para ello una vez tenemos creado el usuario deseado iríamos a **Ajustes > Usuarios y Compañías > Grupos** y una vez ahí filtraríamos por el nombre de usuario como se muestra en la imagen **5.24** y de los grupos que aparecen eliminaríamos el empleado de los que no deseamos. Después de un largo estudio de a que módulos deberían tener acceso se ha decido que puedan acceder a los contactos, al sitio web, al chat y a la Asistencia. Podrán acceder a todos pero no tendrán permisos de edición, simplemente podrán revisar contactos para poder ver su información personal, entrar al sitio web para realizar algún pedido personal, entrar al chat para iniciar conversaciones con algún otro miembro o con los gerentes y acceder a la Asistencia para poder realizar su registro.

Ilustración 5. 24. Permisos grupos

Por último, se han modificado el estilo del Ecommerce para que se adapte al tipo de negocio y también se han creado algunos productos de prueba para poder generar una idea de lo que podría llegar a ser la web como punto de venta de los alimentos. A continuación se puede ver como ha quedado tanto la página principal como la de los productos.

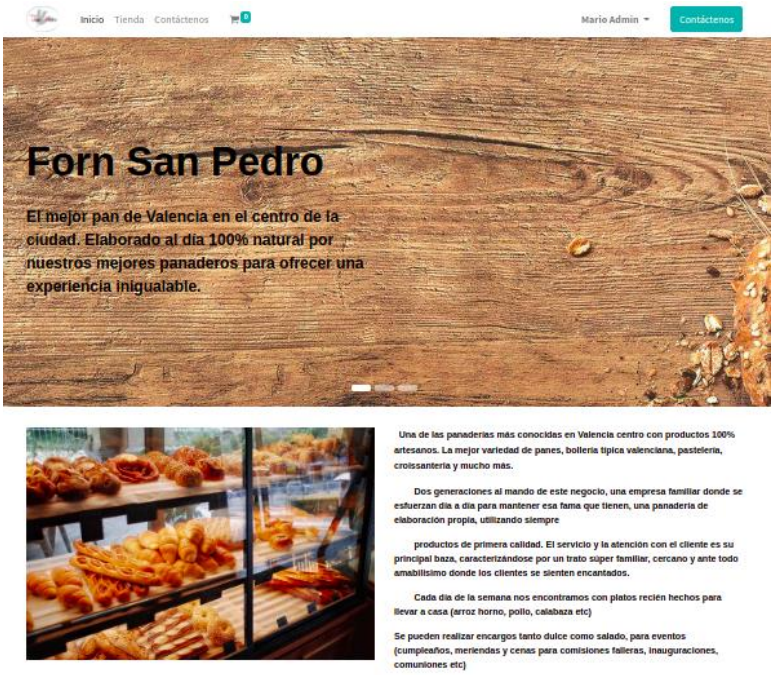


Ilustración 5. 25. Web principal

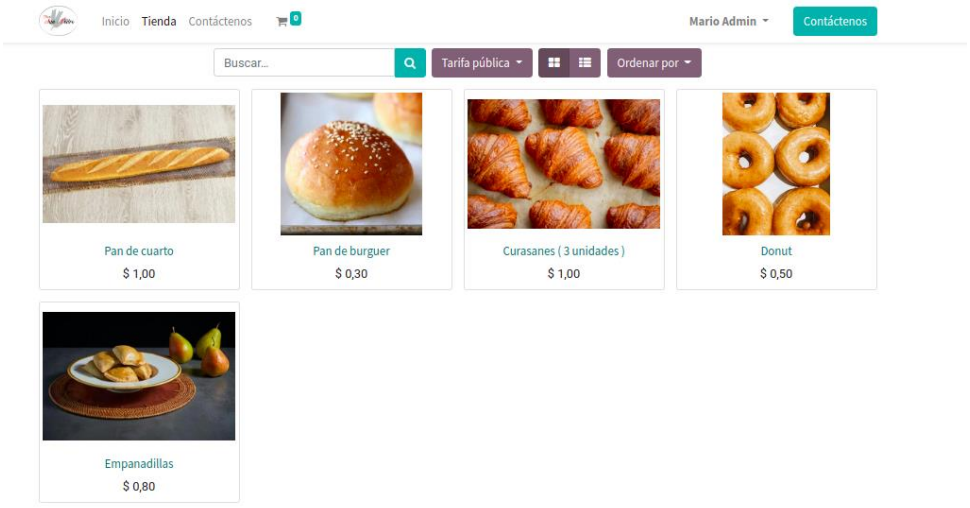


Ilustración 5. 26. Web menú productos

6. Implementación

En este apartado se va a comentar y describir paso por paso el proceso seguido para la realización de la instalación de nuestro despliegue de Odoo. Para este despliegue se utilizará Docker para facilitar tanto la instalación como el lanzamiento de distintas instancias. Este software tiene una serie de características muy interesantes las cuales se explicarán a continuación.

6.1. Que es Docker

En esencia Docker es un software cuya funcionalidad principal es automatizar el despliegue de cualquier aplicación como si fueran contenedores portátiles y autosuficientes que pueden ser ejecutadas en la nube o localmente como es nuestro caso.

Docker empaqueta el software en estas unidades estandarizadas las cuales reciben el nombre de contenedores. Los contenedores incluyen todo lo necesario para que el software pueda ejecutarse, incluidas las herramientas del sistema, el código y las bibliotecas que puede necesitar el programa. De esta forma mediante Docker se puede implementar y adaptar la escala de la aplicación en cualquier sistema con la certeza de que su código se ejecutará .

Como Funciona Docker

Docker se podría considerar como un sistema operativo para contenedores. Su funcionamiento es similar al de una MV (Máquina Virtual) las cuales eliminan la necesidad de administrar directamente el hardware del sistema. De la misma forma los contenedores virtualizan el sistema operativo del sistema. Al instalar Docker en cualquier sistema se puede utilizar para crear, lanzar o detener la aplicación contenida en el contenedor seleccionado. En la imagen siguiente se puede observar cómo funciona y como es la arquitectura de Docker comparada con la de las MV.

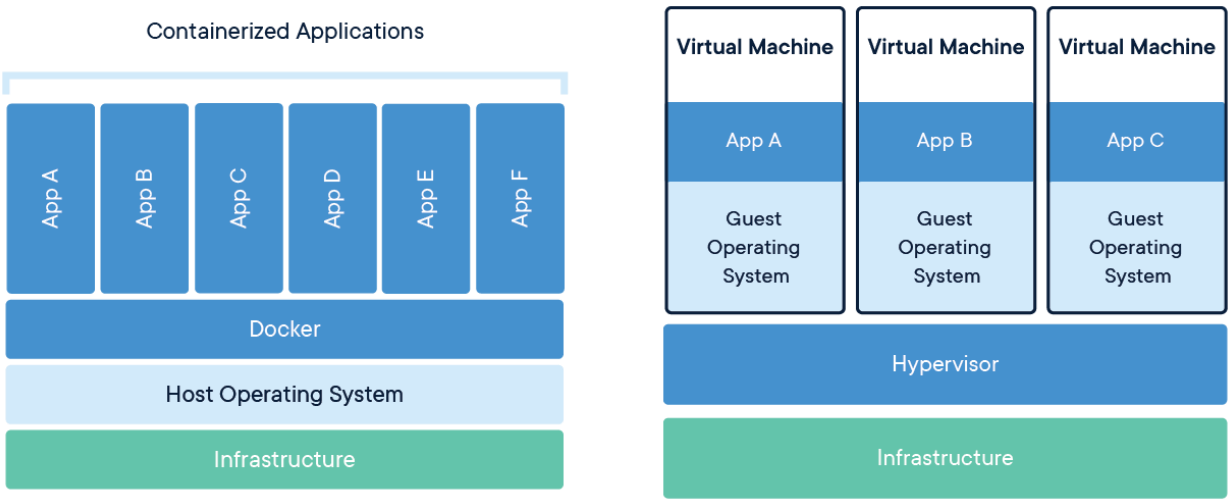


Ilustración 6. 1. Arquitectura Docker (14)

6.2. Implementación de Odoo con Docker

Una vez se ha descrito el sistema que se va a utilizar para la implementación y la puesta en marcha de Odoo, se va a mostrar cómo se ha realizado la instalación de Docker, Odoo y los programas necesarios para el correcto funcionamiento y desarrollo de la aplicación.

Como se ha comentado anteriormente Odoo se encuentra construido mediante el lenguaje Python y utiliza la base de datos PostgreSQL para almacenar los datos por eso tener ambos instalados en el sistema será un requisito básico para poner en funcionamiento el despliegue. Primero se procederá a mostrar las líneas de comandos que se han introducido en la terminal de nuestro sistema para instalarlos.

```
$sudo apt update
$sudo apt install python3.8
$sudo apt-get update
$sudo apt-get install postgresql
```

Una vez se tienen ya los requisitos que necesita Odoo instalados se instalará Docker, Docker-compose y Docker-engine. Acto seguido se reiniciará Docker para que todos los cambios sean tomados correctamente por el sistema como se muestra a continuación:

```
$sudo apt-get install docker.io
$sudo pip install docker-compose
$sudo apt install curl
$sudo curl -sSL https://get.docker.com/ | sh
$sudo service docker restart
```

Con docker instalado, se creará el archivo **docker-compose.yml** en el directorio que se desee que contenga nuestro despliegue, ya que este archivo contiene toda la información necesaria para desplegar nuestro servicio Odoo. Tanto como para crear el directorio como para crear o modificar el archivo se utilizarán los siguientes comandos:

```
$mkdir Odoo14-TFG
$ cd Odoo14-TFG
$nano docker-compose.yml
```

El archivo deberá tener el siguiente formato:

```
version: '3'
services:
  db:
    image: postgres:13
    volumes:
      - db-data:/var/lib/postgresql/data/pgdata
    ports:
      - 5432:5432/tcp
    environment:
    - POSTGRES_PASSWORD=odoo
    - POSTGRES_USER=odoo
    - POSTGRES_DB=postgres
    - PGDATA=/var/lib/postgresql/data/pgdata
  web:
    image: odoo:14.0
    depends_on:
      - db
    ports:
      - "8069:8069/tcp"
    volumes:
      - web-data:/var/lib/odoo
      - ./config:/etc/odoo
      - ./addons:/mnt/extra-addons
volumes:
  db-data:
    driver: local
  web-data:
    driver: local
```

Ilustración 6. 2. Contenido docker-compose.yml

Las partes más importantes del archivo anterior son:

- **image: postgres:13:** Aquí se indicará la imagen de Postgres que se quiere usar, se puede cambiar el valor para asignar la que se necesite, en nuestro caso utilizaremos la 13.
- **environment:** Este campo hace referencia a que nombre de usuario y contraseña tendrá la base de datos Postgres que se va a crear por defecto.
- **image: odoo:14.0:** En este campo se indica la versión Odoo que se va a instalar en el despliegue, en nuestro caso es la versión 14.0 pero se podría seleccionar cualquiera, la 13.0 la 12, etc. Si se quisiera simplemente añadir la última versión disponible de Odoo bastaría con colocar Odoo: latest
- **ports: - "8069:8069/tcp":** En este campo se aginarán los puertos por los que nos conectaremos a Odoo. Si se quisieran tener varias instancias de Odoo corriendo bastaría con modificar el valor del campo y establecer por ejemplo "8070:8069".

Una vez creado nuestro archivo **docker-compose.yml**, se debe lanzar la instancia de Odoo. Para ello basta con ubicarse en el directorio donde se ha creado el archivo antes mencionado y ejecutar:

```
$docker-compose up -d
```

Esto descargará los contenedores docker necesarios, iniciará la base de datos y se podrá acceder a Odoo desde el puerto que se ha seleccionado previamente en el archivo **docker-compose.yml,** en este caso accederíamos al navegador web deseado y accederíamos introduciendo localhost:8069. Una vez realizado esto tocará crear la base de datos, para lo que se elegirá el email, la contraseña de acceso, el lenguaje y el idioma como se puede observar en la figura siguiente.

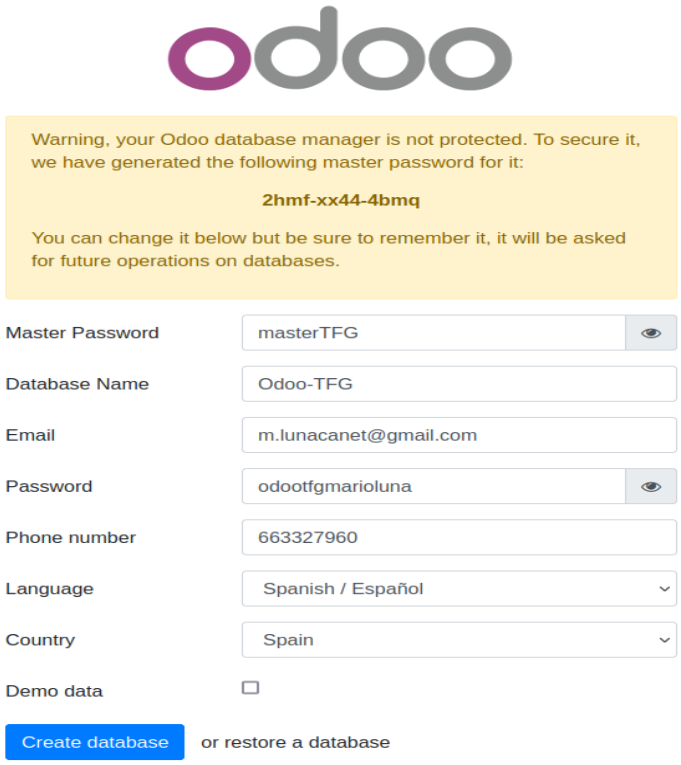


Ilustración 6. 3. Creación base de datos

6.3.Herramientas adicionales

Por último, para poder tener un repositorio al que subir los cambios en caso de fallo del sistema se instalará Git y deberá asociarse nuestro proyecto con un repositorio en GitHub para que almacene las modificaciones. Git es un sistema de control de versiones el cual es gratuito y de código abierto, permite crear ramas y gestionarlas, esto es muy útil cuando hay varias personas trabajando sobre el mismo sistema ya que las modificaciones se realizan sobre las ramas y no afectan al sistema que está en producción. En nuestro caso se utilizará para almacenar las modificaciones desarrolladas en un repositorio en la nube y poder realizar modificaciones sin afectar a la versión funcional que se tiene en producción.

Primero se procederá a instalar git en nuestro sistema y seguidamente se configurarán los detalles de nuestra cuenta de GitHub para que se conecte a ella. Hay que tener en cuenta que el usuario y el email que se van a introducir en los comandos debe ser el mismo que se ha utilizado para crear nuestra cuenta de GitHub. Esto se realizará con los siguientes comandos:

```
$ sudo apt-get install git
$ git config --global user.name "TFGOdoo"
$ git config --global user. Email "m.lunacanet@gmail.com"
```

Una vez ya se tenga git instalado y GitHub configurado y conectado, solo quedará inicializar el repositorio en el directorio en el que se encuentra el despliegue, crear un nuevo repositorio en GitHub y asociarlos. En primer lugar, habrá que crear el directorio en GitHub:

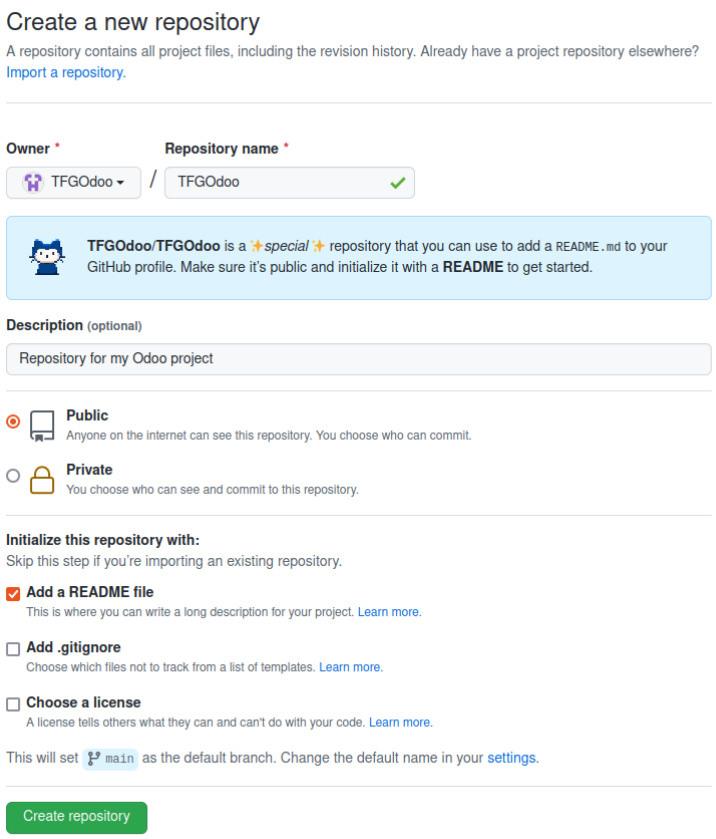


Ilustración 6. 4. Creación repositorio git (15)

Acto seguido se creará el repositorio en nuestro directorio Odoo14-TFG, se asociará con el creado en GitHub y será verificado con los siguientes comandos:

```
$ git init
$ git remote add origin <REMOTE_URL>
$ git remote -v
```

De esta forma ya estaría Git configurado en el sistema para subir todos los cambios que se realicen sobre el despliegue de Odoo. Ahora ya solo faltaría crear algún cambio y probar a subirlo con las siguientes líneas de código:

```
$ git add .  #Añade los archivos en el repositorio local y se preparan para el commit
$ git commit -m "Primer Commit" #Comitea los cambios y los prepara para ser subidos
$ git push origin main #Envía los cambios del repositorio local al repositorio remoto en GitHub
```

Si la instalación es correcta se pedirá la contraseña de GitHub para la subida y mostrará lo siguiente en la consola:

```
root@mario-Z270X-Gaming-K5 /h/m/Odoo14-TFG (master) [128]# git push origin master
Username for 'https://github.com': m.lunacanet@gmail.com
Password for 'https://m.lunacanet@gmail.com@github.com':
Enumerando objetos: 3, listo.
Contando objetos: 100% (3/3), listo.
Compresión delta usando hasta 4 hilos
Comprimiendo objetos: 100% (2/2), listo.
Escribiendo objetos: 100% (3/3), 481 bytes | 481.00 KiB/s, listo.
Total 3 (delta 0), reusado 0 (delta 0)
remote:
remote: Create a pull request for 'master' on GitHub by visiting:
remote:      https://github.com/TFGOdoo/TFGOdoo/pull/new/master
remote:
To https://github.com/TFGOdoo/TFGOdoo.git
 * [new branch]      master -> master
root@mario-Z270X-Gaming-K5 /h/m/Odoo14-TFG (master)#
```

Ilustración 6. 5. Comprobación funcionamiento GitHub

7. Pruebas de funcionamiento

Ahora que el problema ya ha sido analizado y se ha explicado tanto el diseño como el desarrollo de la solución, se debe comprobar el funcionamiento adecuado del sistema y de todo lo desarrollado usando los casos de prueba. Estos casos de prueba se realizan para que todos los requisitos de la aplicación queden revisados y cumplan con las especificaciones planteadas previamente, en concreto se revisarán todos los requisitos específicos que se encuentran en el apartado de Requisitos específicos del sistema del punto **3** de esta memoria.

En primer lugar se comprobará que se cumplen los requisitos comunes a los tres grupos de usuarios. El primero de estos requisitos es el Inicio de Sesión, como se puede ver en los requisitos **3.2**, **3.7** y **3.9**. Se puede observar este requisito en la imagen **7.1** y dependiendo de si el que inicia la sesión es un *Administrador*, un *Cliente* o un *Empleado* tendrá acceso a distintas aplicaciones, cumpliendo así el requisito **3.4** como se puede observar en las imágenes **7.2**, **7.3** y **7.4**.

Ilustración 7. 1. Inicio de Sesión

Ilustración 7. 2. Permisos de acceso del Administrador

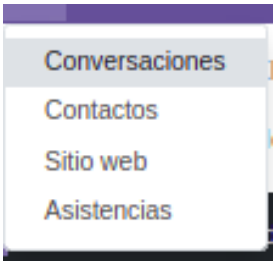


Ilustración 7. 3. Permisos de acceso del Empleado

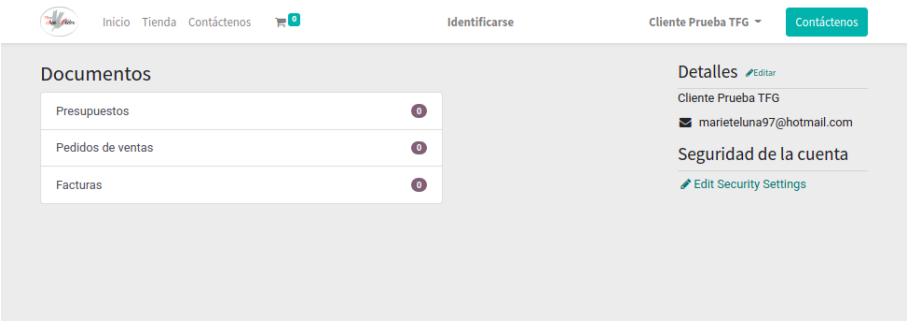


Ilustración 7. 4. Permisos de acceso del Cliente

En segundo lugar se comprobará el funcionamiento del segundo requisito común para todos ellos, el Cierre de Sesión, requisitos **3.3**, **3.8**, **3.11**. Estos requisitos pueden verse comprobados en las imágenes **7.5**, **7.6** y **7.7** respectivamente.

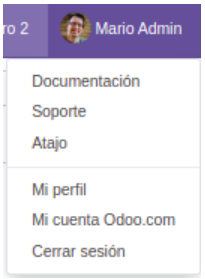


Ilustración 7. 5. Cierre de sesión Administrador

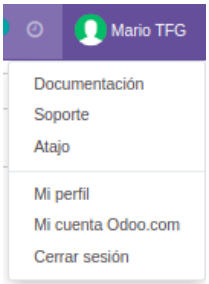


Ilustración 7. 6. Cierre de sesión Empleado

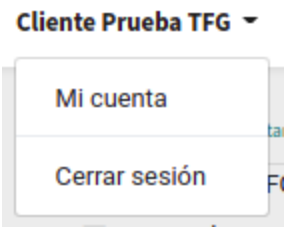


Ilustración 7. 7. Cierre de sesión Cliente

Una vez que ya se han realizado las pruebas para los requisitos comunes, se realizarán las pruebas de los requisitos específicos de cada uno de los grupos de la aplicación, empezando por el grupo de *Administradores*. Este grupo aparte de tener acceso total al sistema será capaz de generar tanto un acuerdo de confidencialidad, requisito **3.5**, como una hoja de reparto, requisito **3.6**. Se puede ver el cumplimiento del primero de estos requisitos en la imagen **7.8**, dónde se puede observar el botón que generará el contrato, y en la imagen **7.9** en donde se ve el acuerdo generado ya cumplimentado. El segundo requisito se puede ver en las imágenes **7.10,** dónde se ve la vista que almacena las hojas de reparto, en la **7.11,** dónde se muestra el formulario en el cual seleccionaremos a los clientes y añadiremos datos y en la **7.12** que muestra la hoja de reparto generada y cumplimentada.

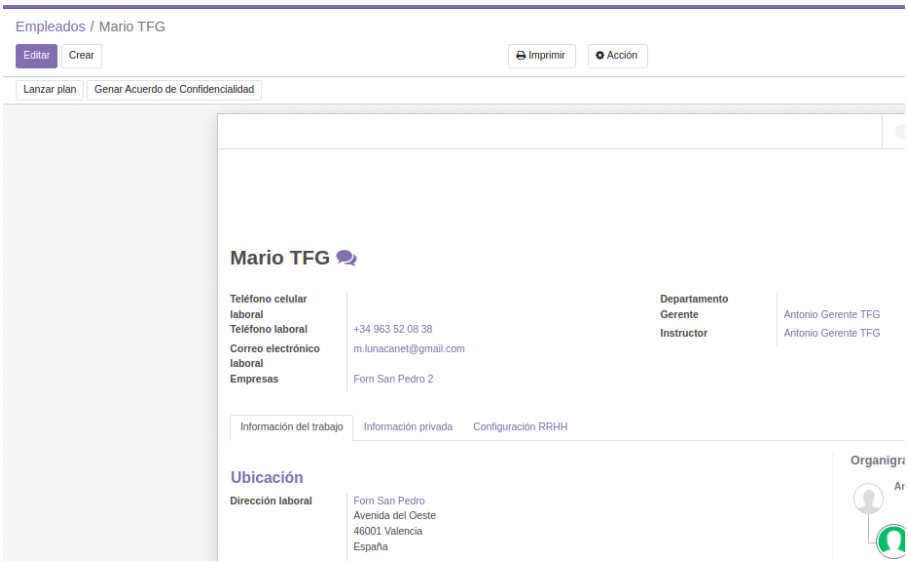
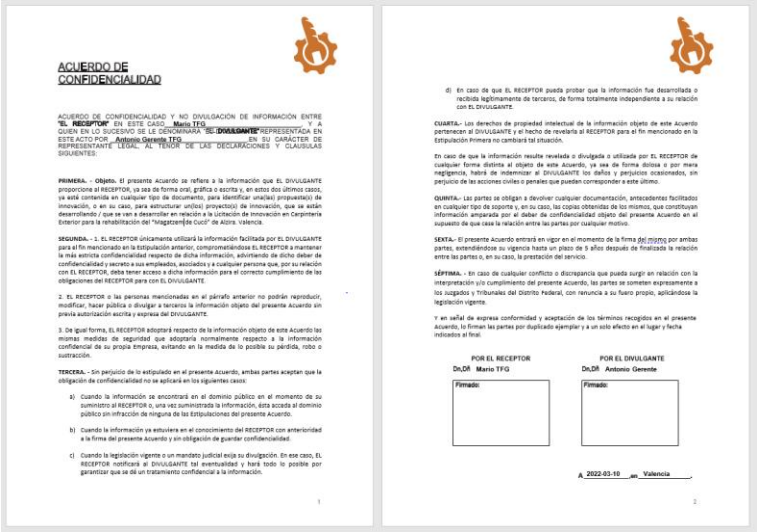


Ilustración 7. 8. Botón generar Acuerdo



ACUERDO DE CONFIDENCIALIDAD

ACUERDO DE CONFIDENCIALIDAD Y NO DIVULGACIÓN DE INFORMACIÓN ENTRE "EL RECEPTOR" EN ESTE CASO, Mario TFG Y A QUIEN EN LO SUCESIVO SE LE DENOMINARA "EL DIVULGANTE" REPRESENTADA EN ESTE ACTO POR Antonio Gerente TFG EN SU CARÁCTER DE REPRESENTANTE LEGAL AL TENOR DE LAS DECLARACIONES Y CLAUSULAS SIGUIENTES:

PRIMERA. - Objeto. El presente Acuerdo ze refiere a la información que EL DIVULGANTE proporcione al RECEPTOR, ya sea de forma oral, gráfica o escrita y, en estos dos últimos casos, ya esté contenida en cualquier tipo de documento, para identificar una(las) propuesta(s) de innovación, o en su caso, para estructurar un(os) proyecto(s) de innovación, que se están desarrollando / que se van a desarrollar en relación a la Licitación de Innovación en Carpintería Exterior para la rehabilitación del "Hogazza de Cucó" de Alzira. Valencia.

SEGUNDA. - 1. EL RECEPTOR únicamente utilizará la información facilitada por EL DIVULGANTE para el fin mencionado en la éstipulación anterior, comprometiéndose EL RECEPTOR a mantener la más estricta confidencialidad respecto de dicha información, advirtiendo de dicho deber de confidencialidad y secreto a sus empleados, asociados y a cualquier persona que, por su relación con EL RECEPTOR, deba tener acceso a dicha información para el correcto cumplimiento de las obligaciones del RECEPTOR para con EL DIVULGANTE.

2. EL RECEPTOR o las personas mencionadas en el párrafo anterior no podrán reproducir, modificar, hacer pública o divulgar a terceros la información objeto del presente Acuerdo sin previa autorización escrita y expresa del DIVULGANTE.

3. De igual forma, EL RECEPTOR adoptará respecto de la información objeto de este Acuerdo las mismas medidas de seguridad que adoptaría normalmente respecto a la información confidencial de su propia empresa, evitando en la medida de lo posible su pérdida, robo o sustracción.

TERCERA. - Sin perjuicio de lo estipulado en el presente Acuerdo, ambas partes aceptan que la obligación de confidencialidad no se aplicará en los siguientes casos:

a) Cuando la información se encontrará en el dominio público en el momento de su suministro al RECEPTOR o, una vez suministrada la información, ésta acceda al dominio público sin infracción de ninguna de las éstipulaciones del presente Acuerdo.

b) Cuando la información ya estuviera en el conocimiento del RECEPTOR con anterioridad a la firma del presente Acuerdo y sin obligación de guardar confidencialidad.

c) Cuando la legislación vigente o un mandato judicial exija su divulgación. En ese caso, EL RECEPTOR notificará al DIVULGANTE tal eventualidad y hará todo lo posible por garantizar que se dé un tratamiento confidencial a la información.

d) En caso de que EL RECEPTOR pueda probar que la información fue desarrollada o recibida legítimamente de terceros, de forma totalmente independiente a su relación con EL DIVULGANTE.

CUARTA.- Los derechos de propiedad intelectual de la información objeto de este Acuerdo pertenecen al DIVULGANTE y el hecho de revelarla al RECEPTOR para el fin mencionado en la Estipulación Primera no cambiará tal situación.

En caso de que la información resulte revelada o divulgada o utilizada por EL RECEPTOR de cualquier forma distinta al objeto de este Acuerdo, ya sea de forma dolosa o por mera negligencia, habrá de indemnizar al DIVULGANTE los daños y perjuicios ocasionados, sin perjuicio de las acciones civiles o penales que puedan corresponder a este último.

QUINTA.- Las partes se obligan a devolver cualquier documentación, antecedentes facilitados en cualquier tipo de soporte y, en su caso, las copias obtenidas de los mismos, que constituyan información amparada por el deber de confidencialidad objeto del presente Acuerdo en el supuesto de que cese la relación entre las partes por cualquier motivo.

SEXTA.- El presente Acuerdo entrará en vigor en el momento de la firma del mismo por ambas partes, extendiéndose su vigencia hasta un plazo de 5 años después de finalizada la relación entre las partes o, en su caso, la prestación del servicio.

SÉPTIMA. - En caso de cualquier conflicto o discrepancia que pueda surgir en relación con la interpretación y/o cumplimiento del presente Acuerdo, las partes se someten expresamente a los Juzgados y Tribunales del Distrito Federal, con renuncia a su fuero propio, aplicándose la legislación vigente.

Y en señal de expresa conformidad y aceptación de los términos recogidos en el presente Acuerdo, lo firman las partes por duplicado ejemplar y a un solo efecto en el lugar y fecha indicados al final.

POR EL RECEPTOR
Dn,Dñ Mario TFG
Firmado:

POR EL DIVULGANTE
Dn,Dñ Antonio Gerente
Firmado:

A 2022-03-10 en Valencia

Ilustración 7. 9. Acuerdo generado

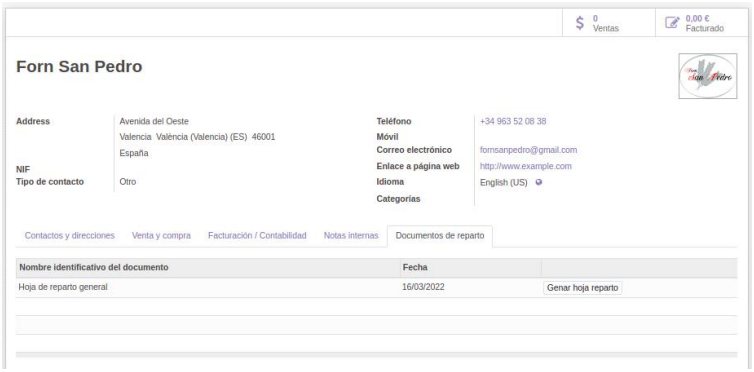
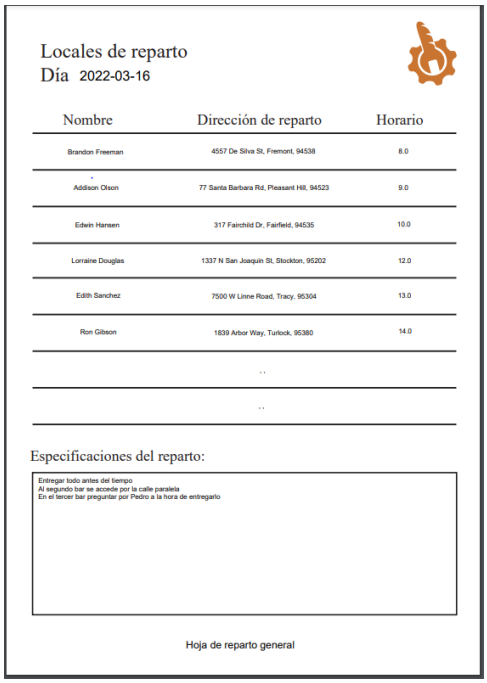


Ilustración 7. 10. Vista de contactos con listado de hojas de reparto

Ilustración 7. 11. Formulario para rellenar datos de la hoja de reparto



Locales de reparto
Día 2022-03-16

Nombre	Dirección de reparto	Horario
Brandon Freeman	4557 De Silva St, Fremont, 94538	8.0
Addison Olson	77 Santa Barbara Rd, Pleasant Hill, 94523	9.0
Edwin Hansen	317 Fairchild Dr, Fairfield, 94535	10.0
Lorraine Douglas	1337 N San Joaquin St, Stockton, 95202	12.0
Edith Sanchez	7500 W Linne Road, Tracy, 95304	13.0
Ron Gibson	1839 Arbor Way, Turlock, 95380	14.0

Especificaciones del reparto:

Entregar todo antes del tiempo
Al segundo bar se accede por la calle paralela
En el tercer bar preguntar por Pedro a la hora de entregarlo

Hoja de reparto general

Ilustración 7. 12. Hoja de reparto generada

En segundo lugar se comprobará si se cumple el requisito específico para el grupo de *Empleados*, el requisito **3.9**. Se puede observar en la imagen **7.13** como el empleado sí que puede acceder a la aplicación seleccionada y registrar su asistencia.

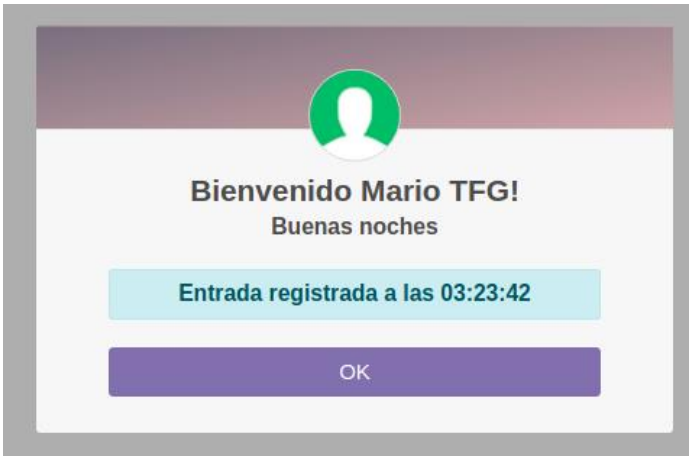


Ilustración 7. 13. Control de asistencia del Empleado

Por último se comprobará que los requisitos del grupo de *Clientes* han sido satisfechos, estos son el **3.12**, el **3.12** y el **3.14**. En la imagen **7.14** se puede ver como el cliente puede editar sus datos, en la imagen **7.15** y **7.16** se puede observar como el cliente puede seleccionar productos y añadirlos al carrito respectivamente y por último en las imágenes **7.17** y **7.18** se puede ver como el cliente es capaz de realizar todo el proceso de compra incluido el pago del producto añadido al carrito.

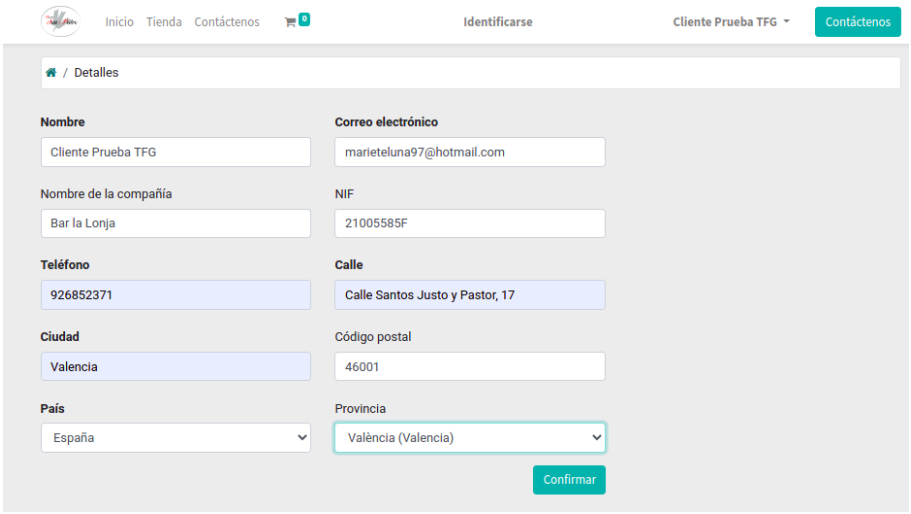


Ilustración 7. 14. Modificación de datos propios por parte del Cliente

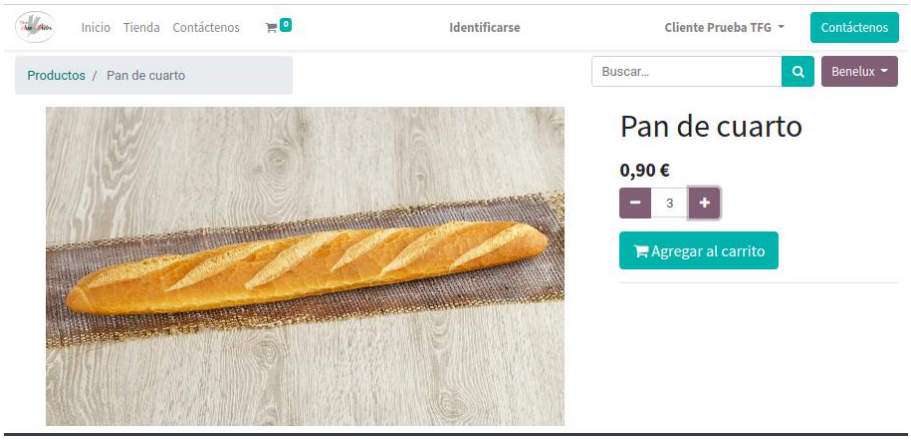


Ilustración 7. 15. Seleccionar productos para añadirlos al carro

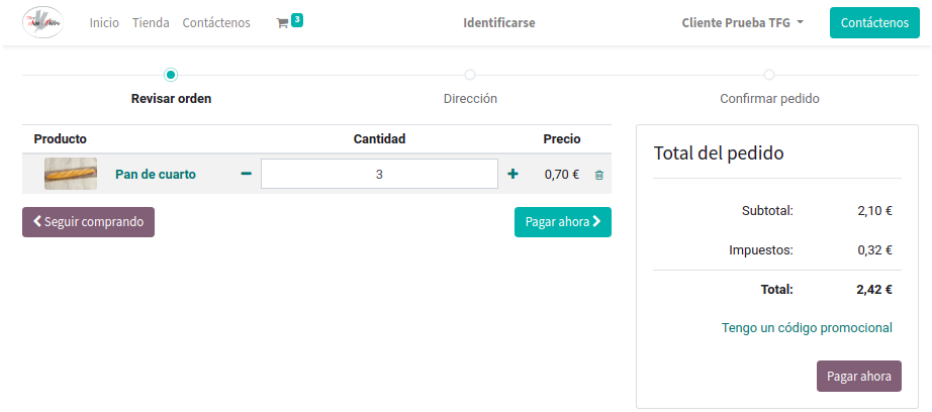


Ilustración 7. 16. Vista del carrito con productos añadidos

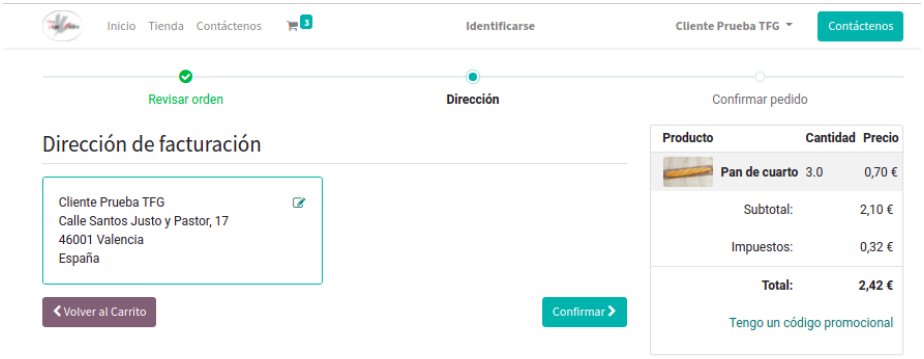


Ilustración 7. 17. Añadir datos de entrega del producto

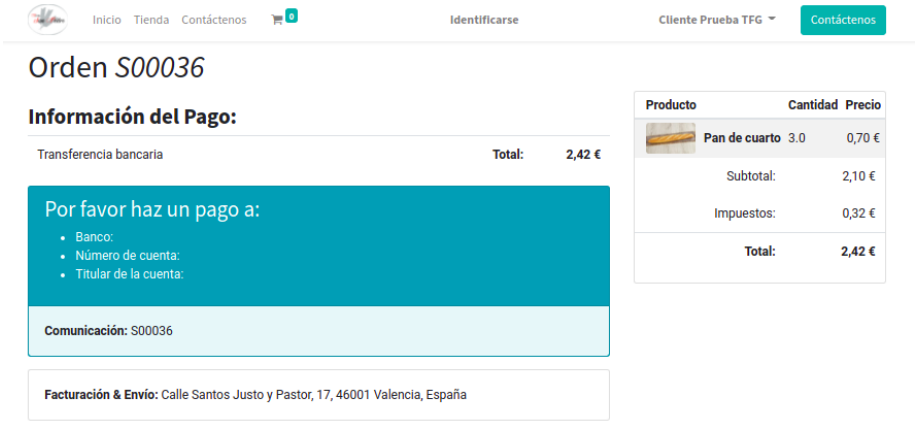


Ilustración 7. 18. Orden de pago con para el pago

8. Conclusiones

Este es el último apartado de la memoria y en él se van a repasar los objetivos planteados en la introducción y se van a plantear los mayores problemas que han ido surgiendo durante la realización de este proyecto.

8.1. Cumpliendo los objetivos

En lo referente a los objetivos planteados al inicio de la memoria, todos han sido completados de manera exitosa. En primer lugar, se ha explicado que son los ERP y como surgieron, en específico el utilizado para el proyecto, Odoo Community. En segundo lugar se ha realizado una instalación completamente funcional de este ERP mediante Docker, el cual también se ha explicado su funcionamiento. En tercer lugar se han seleccionado y configurado los módulos necesarios para agilizar las tareas de la empresa y para poder dotar a la misma de nuevas funcionalidades, como es el portal web o el control de asistencia. Y por último se han podido realizar las modificaciones propuestas de una manera más que satisfactoria. Durante la realización del proyecto y los objetivos he podido ir descubriendo el funcionamiento de Odoo en más profundidad, ya que estaba acostumbrado a esta tecnología pero he descubierto muchísimo más sobre ella y sobre todas las posibilidades que ofrece, ya sea mediante programación o la propia configuración del programa.

8.2. Problemas durante el desarrollo

En esta sección se comentarán los problemas principales que han ido surgiendo durante el desarrollo del proyecto, debido a que ha sido un proyecto distinto a cualquier otro realizado con anterioridad y se han realizado configuraciones de las que se desconocía por completo su existencia y su funcionalidad.

- **Desarrollo de la memoria.** Ya que es la primera vez que he tenido que desarrollar una memoria tan extensa y con una complicación de este nivel, se me ha hecho complicado en algunos momentos saber como continuar y como plasmar el trabajo realizado. También hay que mencionar que debido a mis conocimientos básicos del paquete Office ha habido

ciertas configuraciones que me ha costado un poco implementar, pero debido a la sencillez del programa y la información sobre él no ha sido problema solucionarlos.

- **Instalación de Odoo.** Debido a mis prácticas ya conocía el funcionamiento de Docker en general, pero la instalación de Odoo era realizada de una forma distinta. Averiguar como realizar una instalación independiente y configurarla ha sido un reto que me ha hecho aprender como funciona el programa en su totalidad y las posibilidades que ofrece.
- **Configuración de Odoo.** Ya que en mis prácticas he trabajado de manera extensa con Odoo la programación tanto en Python como en XML no ha sido un verdadero problema. Sin embargo, ya que el sistema sobre el que trabajo ya se encontraba configurado nunca he necesitado realizar configuraciones dentro de Odoo. Por ese motivo la configuración de Odoo desde cero ha supuesto algún que otro problema de una importancia superior.
- **Ecommerce.** Este era un módulo que desconocía por completo y ha supuesto un verdadero reto configurarlo y hacer que funcione como era necesario, sobre todo debido a que los datos de prueba que vienen con una instalación vacía de Odoo han supuesto un problema a la hora de querer eliminar o modificar algunos registros y configuraciones.

Bibliografía

1. Historia del ERP: pasado, presente y futuro | Velneo [Internet]. 2020 [citado 17 de enero de 2022]. [último acceso 12 de enero de 2022]. Disponible en: https://velneo.es/historia-de-erp-pasado-presente-y-futuro/

2. Sistema de planificación de recursos empresariales. En: Wikipedia, la enciclopedia libre [Internet]. 2022 [citado 19 de enero de 2022]. [último acceso 14 de enero de 2022]. Disponible en: https://es.wikipedia.org/w/index.php?title=Sistema_de_planificaci%C3%B3n_de_recursos_empresariales&oldid=141409410

3. apser ER. El software ERP: ejemplos, tipos y uso en la empresa [Internet]. apser - Cloud Computing. 2015 [citado 23 de enero de 2022]. [último acceso 18 de enero de 2022]. Disponible en: https://apser.es/el-software-erp-ejemplos-tipos-y-uso-en-la-empresa/

4. roro. Dilema: escoger un ERP horizontal o vertical [Internet]. SEMIC. 2016 [citado 10 de febrero de 2022]. [último acceso 3 de febrero de 2022]. Disponible en: https://www.semic.es/es/content/dilema-escoger-un-erp-horizontal-o-vertical

5. TI RB. Los 4 mejores ERP para pymes en 2021: análisis comparativo | SAP [Internet]. 2021 [citado 10 de febrero de 2022]. [último acceso 4 de febrero de 2022]. Disponible en: https://revistabyte.es/noticias/los-4-mejores-erp-para-pymes-en-2021-analisis-comparativo/

6. Oracle ERP: Los Mejores 3 Módulos de ERP [Internet]. [citado 10 de febrero de 2022]. [último acceso 4 de febrero de 2022]. Disponible en: https://www.chetu.com/es/blogs/technical-perspectives/oracle-erp-modules.php

7. Microsoft Dynamics 365 - Qué es y para qué sirve [Internet]. Canal ERP. 2019 [citado 10 de febrero de 2022]. [último acceso 5 de febrero de 2022]. Disponible en: https://canalerp.com/microsoft-dynamics-365-que-es-y-para-que-sirve/

8. Portal TIC. Workday: ¿Es el software de RRHH adecuado para multinacionales? [Internet]. TIC Portal. [citado 10 de febrero de 2022]. [último acceso 7 de febrero de 2022]. Disponible en: https://www.ticportal.es/temas/software-gestion-recursos-humanos/programas-recursos-humanos/workday

9. Welcome to Python.org [Internet]. Python.org. [citado 7 de marzo de 2022]. [último acceso 17 de febrero de 2022]. Disponible en: https://www.python.org/

10. Group PGD. PostgreSQL [Internet]. PostgreSQL. 2022 [citado 7 de marzo de 2022]. [último acceso 18 de febrero de 2022]. Disponible en: https://www.postgresql.org/

11. Odoo Enterprise vs Community | Odoo Editions Comparison [Internet]. Odoo S.A. [citado 8 de febrero de 2022]. [último acceso 3 de febrero de 2022]. Disponible en: https://www.odoo.com/page/editions

12. ieee830.pdf [Internet]. [citado 18 de febrero de 2022]. [último acceso 13 de febrero de 2022]. Disponible en: https://www.fdi.ucm.es/profesor/gmendez/docs/is0809/ieee830.pdf

13. Building a Module — documentación de Odoo - 14.0 [Internet]. [citado 7 de marzo de 2022]. [último acceso 27 de febrero de 2022]. Disponible en: https://www.odoo.com/documentation/14.0/es/developer/howtos/backend.html

14. Wikimedia Snapshot [Internet]. [citado 7 de marzo de 2022]. [último acceso 7 de marzo de 2022]. Disponible en: https://commons.wikimedia.org/wiki/File:Docker-containerized-and-vm-transparent-bg.png

15. GitHub: Where the world builds software [Internet]. GitHub. [citado 7 de marzo de 2022]. [último acceso 7 de marzo de 2022]. Disponible en: https://github.com/

Apéndice A – Acrónimos

ERP: Enterprise Resource Planning.

XML: Extensible Markup Language.

MV: Máquina Virtual.

CSS: Cascading Style Sheets.

PDF: Portable Document Format.

MVC: Model View Controller o Modelo Vista Controlador.

ERS: Especificación de Requisitos Software.

IEEE: Institute of Electrical and Electronics Engineers.

W3C: World Wide Web Consortium.

OCA: Odoo Community Association.

CEO: Chief Executive Officer.

RRHH: Recursos Humanos.

CRM: Customer Relationship Management.

SAP: Systems, Applications, Products in Data Processing.

SaaS: Software as a Service.

MRP: Manufacturing Resource Planning.

IBM: International Business Machines

IMC: Inventory Management and Control.

BOM: Bill of Materials.

ODS: Objetivos de desarrollo sostenible.

Apéndice B

- Objetivos de desarrollo sostenible (ODS)

OBJETIVOS DE DESARROLLO SOSTENIBLE

Grado de relación del trabajo con los Objetivos de Desarrollo Sostenible (ODS).

Objetivos de Desarrollo Sostenibles	**Alto**	**Medio**	**Bajo**	**No Procede**
ODS 1. **Fin de la pobreza.**				**X**
ODS 2. **Hambre cero.**				**X**
ODS 3. **Salud y bienestar.**				**X**
ODS 4. **Educación de calidad.**				**X**
ODS 5. **Igualdad de género.**				**X**
ODS 6. **Agua limpia y saneamiento.**				**X**
ODS 7. **Energía asequible y no contaminante.**				**X**
ODS 8. **Trabajo decente y crecimiento económico.**	**X**			
ODS 9. **Industria, innovación e infraestructuras.**		**X**		
ODS 10. **Reducción de las desigualdades**.				**X**
ODS 11. **Ciudades y comunidades sostenibles**.				**X**
ODS 12. **Producción y consumo responsables.**			**X**	
ODS 13. **Acción por el clima.**				**X**
ODS 14. **Vida submarina.**				**X**
ODS 15. **Vida de ecosistemas terrestres.**				**X**
ODS 16. **Paz, justicia e instituciones sólidas.**				**X**
ODS 17. **Alianzas para lograr objetivos.**				**X**

Reflexión sobre la relación del TFG/TFM con los ODS y con el/los ODS más relacionados.

Este proyecto se centra en la optimización y mejora de una pequeña empresa de un sector especializado en la venta de productos de primera necesidad, en este caso es una pequeña panadería tradicional. Por este motivo no se encuentra relacionado con ninguno de los "principales" ODS que se muestran en la tabla de más arriba. Si que se encuentra relacionado sin embargo con los ODS número 8, 9 Y 10 respectivamente y en los siguientes párrafos se explicará de que manera se encuentra relacionado.

En primer lugar, respecto al ODS número 8, el cual hace referencia a el trabajo decente y el crecimiento económico, podemos ver que este proyecto se encuentra altamente relacionado con ella. En primer lugar, respecto al trabajo decente, este proyecto de ser llevado a cabo y aplicarse en el mundo real, podría llegar a quitar horas y horas de trabajo para las personas que se encargan de toda la gestión de este tipo de empresas. Conozco de primera mano el sacrificio que conlleva tener un negocio de este tipo, tanto en dinero perdido por procesos ineficientes como en el tiempo de trabajo de horas extra que conlleva recabar los pedidos y tener que organizarlos para el día siguiente, y poder implementar un despliegue de este tipo conseguiría reducir todos esos costes y horas de trabajo de una manera radical. Respecto al crecimiento económico: gracias a este proyecto, se podría crear una web de manera super sencilla lo cual serviría para promocionarse en la red y conseguir nuevos clientes y compradores y además se podrían internalizar procesos que son contratados externamente ahorrando costes lo que aumentaría a su vez las ganancias de la empresa.

En segundo lugar, encontramos el ODS número 9 relacionado con la industria, innovación y la infraestructura. Este ODS también se encuentra relacionado con nuestro proyecto en primer lugar porque afectaría a la industria tanto de los pequeños comercios como de las panaderías y pastelerías y en segundo lugar se encuentra relacionado con la parte de la innovación ya que la instalación de este despliegue conllevaría un avance tecnológico increíble para un tipo de negocio que lleva gestionándose de la misma forma desde hace décadas. Por último, el proyecto también serviría para gestionar de una manera más eficiente la infraestructura de cualquier tipo de pequeña empresa, pero más en concreto de las empresas que pertenecen a este sector en concreto.

Y por último se debe mencionar la relación que existe entre este proyecto y el ODS número 12 que trata sobre la producción y el consumo responsables. Aunque es el que se encuentra relacionado en menor medida con este proyecto si que podemos identificar una conexión real entre las dos partes. Debido a la implementación de este sistema en cualquier negocio de tipo pastelería o panadería se podría gestionar de manera responsable la producción de alimentos gracias al estudio de los registros de ventas. De esta forma se podría realizar una aproximación de los productos necesarios al día para no producir un exceso que acabaría desperdiciándose. También serviría para identificar el consumo de materias primas para utilizar solo las necesarias y como se ha mencionado con anterioridad no generar un exceso que lleve a su desperdicio.