



UNIVERSITAT  
POLITÈCNICA  
DE VALÈNCIA



Escola Tècnica  
Superior d'Enginyeria  
Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica  
Universitat Politècnica de València

# DESARROLLO DE UNA APLICACIÓN PARA ODOO ERP

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

*Autor:* Pau Sastre Pons

*Tutor:* José Vicente Busquets Mataix

Curso 2019-2020



# Resumen

Desde la aparición de los primeros computadores la tecnología ha supuesto un cambio importante en el funcionamiento de las empresas y de la sociedad en general. Ha automatizado procesos que han permitido a las empresas crecer, e incluso otras formas de negocio que anteriormente ni se podían imaginar. Existen muchos tipos de Sistemas de Información, y uno de los más utilizados son los sistemas ERP. Estos sistemas nos ayudan a controlar la producción de la empresa, manteniendo inventarios y flujos de información, almacenando y relacionando todas las partes (clientes, proveedores y trabajadores).

En este proyecto vemos qué es un ERP, sus principales alternativas, y nuestra elección: Odoo, sobre el cual se desarrolla una aplicación. Nuestra aplicación sirve para automatizar el funcionamiento de una PYME deportiva, con ello veremos como instalar, implementar y configurar un sistema Odoo con una aplicación personalizada a medida para un uso real.

**Palabras clave:** Sistemas de Información, Sistema ERP, Odoo, PYME.

---

# Resum

Des de l'aparició dels primers ordinadors la tecnologia ha suposat un canvi important en el funcionament de les empreses i de la societat en general. Ha automatitzat processos que han permès créixer, i inclús hi han sortit altres formes de negoci que anteriorment ni es podien imaginar. Existeixen molts tipus de Sistemes d'Informació, i un dels més utilitzats són els sistemes ERP. Aquests sistemes ens ajuden a controlar la producció de l'empresa, mantenint inventaris i fluxos d'informació, emmagatzemant i relacionant totes les parts (clients, proveïdors i treballadors).

En aquest projecte veiem què és un ERP, les seues principals alternatives, i la nostra elecció: Odoo, sobre la qual es desenvolupa una aplicació. La nostra aplicació serveix per a automatitzar el funcionament d'una PYME esportiva, amb açò veurem com instal·lar, implementar i configurar un sistema Odoo amb una aplicació personalitzada a mesura per a l'ús real.

**Paraules clau:** Sistemes d'Informació, Sistema ERP, Odoo, PYME.

---

# Abstract

Since the first computer appeared technology has meant an important change in the way companies and overall society work. It has automated processes that have allowed companies to grow, and even other forms of business that previously no one could not be imagined. There are many types of Information Systems, and one of the most used is ERP systems. These systems help us control the production of a company, maintaining inventories and information flows, storing and relating all parts (clients, suppliers and workers).

In this project we see what is an ERP, its main alternatives, and our choice: Odoo, on which we developed an application. Our application is used to automate the operation of a sports PYME, with this we will see how to install, implement and configure an Odoo system with a custom application developed for real use.

**Key words:** Information Systems, RP systems, Odoo, PYME.

---



# Índice general

---

<b>Índice general</b>	VII
<b>Índice de figuras</b>	IX
<b>Índice de tablas</b>	X
<hr/>	
<b>1 Introducción</b>	<b>1</b>
1.1 Motivación . . . . .	1
1.2 Objetivos . . . . .	2
1.3 Estructura de la memoria . . . . .	2
<b>2 Estado del arte</b>	<b>5</b>
2.1 ¿Qué es un ERP? . . . . .	5
2.2 Principales ERP . . . . .	7
2.3 Elección de Odoo . . . . .	11
2.4 Historia de Odoo . . . . .	12
2.5 Ediciones de Odoo . . . . .	14
2.6 Arquitectura . . . . .	14
<b>3 Análisis</b>	<b>17</b>
3.1 Especificación de requisitos software . . . . .	17
3.1.1 Introducción . . . . .	17
3.1.2 Descripción general . . . . .	18
3.1.3 Requisitos específicos . . . . .	20
3.2 Ejemplo de uso concreto . . . . .	27
<b>4 Diseño de la solución</b>	<b>31</b>
4.1 Diagramas de casos de uso . . . . .	31
4.2 Diagrama de clases . . . . .	35
4.3 Prototipado . . . . .	36
4.3.1 Vistas Clientes . . . . .	37
4.3.2 Vistas Trabajadores . . . . .	39
4.3.3 Vistas Servicios . . . . .	42
<b>5 Desarrollo</b>	<b>45</b>
5.1 Modelos . . . . .	46
5.2 Vistas . . . . .	48
5.3 Seguridad . . . . .	51
5.4 Otros archivos . . . . .	54
<b>6 Implantación y mantenimiento</b>	<b>57</b>
6.1 Configuración general . . . . .	57
6.2 Configuración del servicio . . . . .	58
6.3 Configuración del servidor inverso . . . . .	59
6.4 Configuración del HTTPS . . . . .	60
6.5 Copia de seguridad y gestión de bases de datos . . . . .	61
6.6 Instalación y actualización de módulos . . . . .	62
6.7 Actualización de servidor . . . . .	63
<b>7 Pruebas funcionales</b>	<b>65</b>

---

<b>8 Conclusiones y trabajos futuros</b>	<b>75</b>
<b>Bibliografía</b>	<b>77</b>

---

Apéndices

<b>A Configuración del sistema</b>	<b>79</b>
A.1 Máquina Virtual . . . . .	79
A.2 Instalación y configuración de Odoos 12.0 . . . . .	79
A.3 Otras herramientas . . . . .	81
A.3.1 Terminator . . . . .	82
A.3.2 PyCharm . . . . .	82
A.3.3 Pencil . . . . .	82
A.3.4 Lucidchart . . . . .	82
<b>B Acrónimos</b>	<b>83</b>



# Índice de figuras

---

2.1	Gráfico de los motivos de elección ERP	7
2.2	Gráfico de los beneficios de los ERPs	7
2.3	Tipos ERPs en 2017	8
2.4	Tipos ERPs en 2018	9
2.5	Gráfico de la elección de ERPs	11
2.6	Figura de la arquitectura básica de Odoos	15
3.1	Horario cursos de natación	29
3.2	Horario cursos de pádel	29
3.3	Horario cursos de tenis	29
3.4	Horario cursos de artes marciales	30
4.1	Diagrama de casos de uso 1	32
4.2	Diagrama de casos de uso 2	33
4.3	Diagrama de casos de uso 3	34
4.4	Diagrama de casos de uso 4	35
4.5	Diagrama de clases	36
4.6	Boceto vista principal	37
4.7	Boceto clientes	37
4.8	Boceto nuevo cliente	38
4.9	Boceto detalles cliente	39
4.10	Boceto trabajadores	40
4.11	Boceto nuevo trabajador	40
4.12	Boceto detalles trabajador	41
4.13	Boceto servicios	42
4.14	Boceto nuevo servicio	43
4.15	Boceto detalles servicio	44
5.1	Directorios módulo	45
5.2	Usuario Administrador	54
5.3	Aplicación SportService	55
6.1	Boceto detalles servicio	62
6.2	Boceto detalles servicio	62
7.1	Inicio de sesión	65
7.3	Menú principal	65
7.2	Cierre de sesión	66
7.4	Vista Clientes	66
7.5	Vista Trabajadores	67
7.6	Vista Servicios	67
7.7	Vista nuevo cliente	68
7.8	Detalles cliente	69
7.9	Vista nuevo trabajador	69

7.10	Detalles trabajador	70
7.11	Vista nuevo servicio	71
7.12	Detalles servicio	72
7.13	Notificación alta	72
7.14	Notificación alta	73
7.15	Notificación alta	73
7.16	Notificación baja	73
A.1	Terminal correcto	80
A.2	Navegador correcto	81
A.3	Modo desarrollador	81

## Índice de tablas

---

2.1	Historial versiones de Odoo	13
3.1	Requisito específico 01	20
3.2	Requisito específico 02	21
3.3	Requisito específico 03	21
3.4	Requisito específico 04	21
3.5	Requisito específico 05	22
3.6	Requisito específico 06	22
3.7	Requisito específico 07	23
3.8	Requisito específico 08	23
3.9	Requisito específico 09	24
3.10	Requisito específico 010	24
3.11	Requisito específico 11	25
3.12	Requisito específico 12	25
3.13	Requisito específico 13	26
3.14	Requisito específico 14	26
3.15	Requisito específico 15	27
3.16	Requisito específico 16	27
5.1	Tabla resumen de permisos	53

---

---

# CAPÍTULO 1

## Introducción

---

La introducción es el primer apartado de la memoria, a su vez se divide en tres partes: motivación, objetivos y estructura de la memoria. La motivación explica las razones por las que se desarrolla el proyecto, tanto académicos como personales. Los objetivos enumera los objetivos que debe implementar la aplicación una vez desarrollada, a grandes rasgos el alcance del proyecto. Por último la estructura de la memoria lista y describe brevemente cada capítulo de la memoria.

### 1.1 Motivación

---

En la actualidad cualquier empresa de cualquier ámbito hace uso de la tecnología, esta impacta sobre ellas, acelera su crecimiento, mejora su eficiencia, e incluso crea industrias nuevas. Este fenómeno está muy asimilado en el año 2020, pero todavía hay muchas empresas, sobretodo pymes, con poca o nula implementación tecnológica. Muchas de estas todavía gestionan sus datos de forma manual, añadiendo simplemente un ordenador con programas básicos como navegador, ofimática, correo electrónico, etc, mejoraría mucho su eficiencia. Si añadimos un software específico de gestión de recursos, con herramientas adaptadas a las necesidades de la empresa, se mejora todavía más la eficiencia, y añade nuevas funcionalidades que mejorarían la empresa en todos sus espacios. Buena parte de esta filosofía es la que define la tecnología Odoo ERP, que se define más adelante [2](#).

Personalmente tengo interés en aprender un ERP, ya que es una tecnología informática ampliamente utilizada y que no he visto durante todo el grado. Como he comentado en el párrafo anterior, también es interesante poder cubrir de esta forma las necesidades de muchas pymes, así que considero este proyecto un aprendizaje útil para proyectos reales.

Por último, comentar que me considero afín a la filosofía software libre (<https://www.gnu.org/philosophy/free-sw.es.html>). Para el desarrollo de este proyecto se ha utilizado software libre al completo (en un sistema operativo Linux, memoria redactada en LaTeX, así como más software complementario). Odoo es el principal ERP con opción de software libre, detallado en el apartado [2.5](#), que a su vez se implementa utilizando tecnologías basadas en el software libre (Python, PostgreSQL y XML). Por lo tanto encaja bien con esta tecnología, ya que no se trata de software privativo como la mayoría de ERPs y mi proyecto al completo se puede basar en esta filosofía.

## 1.2 Objetivos

---

El principal objetivo de este trabajo de fin de grado es desarrollar una aplicación mediante el uso de la tecnología Odoo ERP. El objetivo de la aplicación es gestionar actividades deportivas de una empresa, que se detalla en el apartado 3. A continuación se enumera una serie de objetivos sin los que sería imposible alcanzar dicha meta final:

- Entender que es un ERP, Odoo, y su utilidad.
- Instalar y configurar correctamente un sistema Odoo.
- Planificar y desarrollar la aplicación utilizando las metodologías de desarrollo ágil de software y de desarrollo centrado en el usuario.
- Por último, a nivel personal, desarrollar una aplicación real combinando todo lo anterior y todo lo aprendido durante el grado de ingeniería informática.

## 1.3 Estructura de la memoria

---

A continuación se describen brevemente los ocho capítulos en los que se divide la memoria del proyecto:

- **Capítulo 1, Introducción:** apartado en el que nos encontramos. Plantea objetivos, motivaciones y estructura del proyecto.
- **Capítulo 2, Estado del arte:** se explica que es un sistema ERP, así como el contexto actual de la tecnología. También se explica qué diferencia Odoo y el porqué de su elección, sus versiones, historia y arquitectura.
- **Capítulo 3, Análisis:** se plantea el caso de estudio mediante una especificación de requisitos software según el estándar IEEE 830 y un posterior ejemplo concreto.
- **Capítulo 4, Diseño de la solución:** mediante metodologías de desarrollo ágil de software y de desarrollo centrado en el usuario se plantea la solución al problema.
- **Capítulo 5, Desarrollo de la solución:** aquí vemos plasmado el desarrollo mostrado la estructura de un módulo Odoo. Explicamos todos sus subdirectorios y archivos, destacando los modelos desarrollados, las vistas y la configuración de seguridad. Los modelos implementan tanto la persistencia como la lógica, y las vistas la interfaz de usuario.
- **Capítulo 6, Implantación y mantenimiento:** despliegue y configuración de Odoo sobre Ubuntu, uso de un proxy inverso, HTTPS, copias de seguridad y actualización del servidor y sus módulos.
- **Capítulo 7, Pruebas funcionales:** conjunto de pruebas que se realizan sobre la aplicación desarrollada, para comprobar errores de funcionamiento y seguridad, así como comprobar que se cumplen los requisitos planteados.
- **Capítulo 8, Conclusiones y trabajos futuros:** cierre del proyecto, se comentan los resultados obtenidos en relación a los objetivos propuestos. Aparte se comentan posibles mejoras a implementar en la aplicación, que no se incluye en el proyecto actual por motivos de tiempo o conocimientos.

Además la memoria contiene una biografía y tres apéndices:

- 
- **Apéndice A, Configuración del sistema:** configuración para el desarrollo del proyecto, incluyendo máquina virtual, sistema Odoo, IDE, y otras herramientas usadas.
  - **Apéndice B, Acrónimos:** definición de los múltiples acrónimos que aparecen en la memoria.



---

---

## CAPÍTULO 2

# Estado del arte

---

El segundo apartado de la memoria nos introduce la tecnología usada para el desarrollo de la aplicación, comentando la situación actual de esta. En este apartado empezamos con la definición y principales características de los ERP, así como los diferentes software ERP en el mercado actual. Tras explicar de que se trata nos centraremos en Odoo: el porqué de su elección, se explica brevemente su historia y ediciones, y finalmente una descripción de la arquitectura.

### 2.1 ¿Qué es un ERP?

---

Las siglas en inglés significan *Enterprise Resource Planning*, que traducido al español sería Planificación de recursos de la empresa. Históricamente es una evolución de los software y sistemas de planificación de los materiales, las siglas MRP en inglés (*Material Requeriment Planning*), cuyos objetivos básicos eran tres:

- Control de productos y materiales para producción y entrega a clientes.
- Tener siempre el nivel correcto en los inventarios.
- Estructurar periódicamente la fabricación, entrega y actividades de compra.

Los primeros sistemas de planificación fueron desarrollados por el gobierno de los Estados Unidos de América durante la segunda guerra mundial, con el objetivo de gestionar material bélico. Pero no fue hasta finales de los años 50 que se empezó a utilizar en el mundo empresarial. Algunas empresas empezaron a incorporar estos sistemas para mejorar la productividad y organización de las mismas, añadiendo así otras funcionalidades, mayoritariamente de administración económica. Más adelante, en los años 80, evolucionó a MRP II, *Manufacturing Requeriment Planning* en inglés. Esta evolución incorpora más funcionalidades como los empleados o finanzas más avanzadas. Por último sobre los años 90 se incorporan otras áreas de la empresa aparece el término ERP, evolucionando a su vez hacia modelos de suscripción por el uso del servicio, también llamado SaaS o *cloud computing*.

Una de las primeras definiciones viene dada por *Dave Swartz*, que habla de los ERPs como una solución de software que integra información y procesos de negocio en torno a una Base de Datos compartida por toda la organización (Dave Swartz, 2000).

Según *Soroka* un sistema ERP es un paquete de programas estandarizados que le permite a una compañía automatizar e integrar la mayor parte de sus procesos de negocios, compartir datos y prácticas entre todos los miembros de la organización, y producir y acceder a la información en un ambiente de tiempo real (A J Soroka, 2002).

Una vez claros los orígenes y el concepto de ERP pasamos a sus principales objetivos en la empresa. En primer lugar la optimización de los procesos empresariales, intentando abarcar su totalidad. También, se necesita acceder a toda la información de la empresa, y compartir esta información entre sus partes. Por último eliminar datos y operaciones innecesarias.

Existen muchas características que definen los ERPs, algunas de ellas dependen del ERP elegido, pero hay dos principales que comparten todos:

- **Modulares.** Incorpora distintos módulos para gestionar los departamentos de la empresa. Como ya se ha comentado, la base de datos es común, por lo tanto esta todo conectado para facilitar el intercambio de información entre secciones de la empresa. Por ejemplo un módulo puede gestionar el departamento de ventas, que a su vez comparte información necesaria con otro módulo de organización de almacenes.
- **Configurables.** Cada empresa es diferente, y los ERP se pueden modificar adaptándose así a las necesidades de cada empresa. Estas necesidades son específicas de cada empresa, y a su vez pueden variar con el tiempo. Un ejemplo común es un módulo de gestión de almacenes, que para empresa se tendría que adaptar a su tamaño de inventario, número de trabajadores, horarios, tipo de productos, y un largo etcétera.

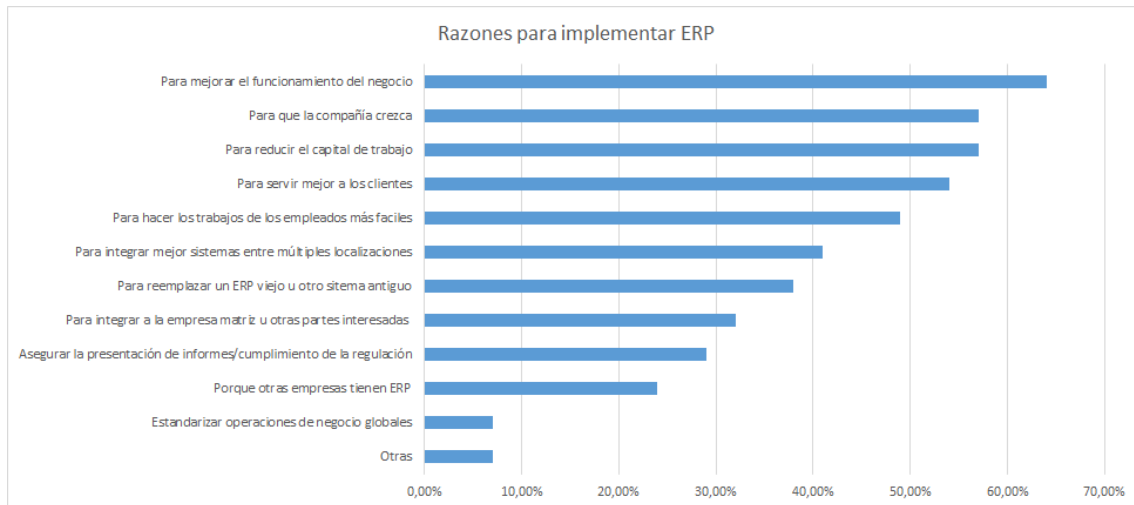
Implementar un ERP en una compañía comporta una serie de ventajas respecto a otro tipo de soluciones. Muchas veces las empresas tienen una aplicación cerrada para cada parte de esta, algo que suele complicar la gestión. En cambio si usamos un ERP se automatizan todos los procesos de la empresa y se integran todas las bases de datos en una plataforma, mejorando así la comunicación interna entre departamentos. Así mismo también se automatizan los procesos de la empresa, y se ahorran tiempo y costes debido a que la planificación aumenta la rentabilidad. Como hemos comentado antes la flexibilidad e los ERP permite una optimización para cada caso, entonces se cubre cualquier particularidad de cada empresa. Hay soluciones relativamente baratas, dependiendo del nivel de personalización que se desee, pero teniendo en cuenta que puede cubrir todas las áreas de la empresa esto supone un ahorro respecto a desarrollar distinto software para cada departamento.

Como todo sistema de información, los ERPs también tienen inconvenientes que hay que comentar. Un problema común es tras su puesta en marcha en una empresa el personal debe aprender a utilizar el software. Esto implica educar al personal, así como posibles errores por desconocimiento, o pérdidas de tiempo, lo que se traduce al final en pérdidas económicas. Si bien antes comentábamos que puede ser una solución barata, dependiendo del nivel de personalización puede llegar a ser un inconveniente, si se trata de una empresa que requiere de mucha especialización. Otros factores que puede incrementar el coste son la instalación del software, su actualización y mantenimiento o licencias, que depende del tipo de software y del contrato para dicho desarrollo. Al implementar un ERP se debe de tener en cuenta el hardware y infraestructura para un óptimo funcionamiento, dado el caso de que haya que renovar el sistema informático de la empresa tendríamos un incremento indirecto en el coste. Esto último, se puede solventar mediante sistemas en la nube, que se comentará más adelante. Por último, en algunas ocasiones los proyectos son más complejos técnicamente que la solución real que necesita el cliente, sobre todo proyectos para pequeñas empresas que requieran funcionalidades más sencillas pueden sobrar componentes.

A continuación, en la figura 2.1 podemos ver las principales razones por las cuales una empresa decide implementar un ERP. Los datos proceden de una encuesta llevada a

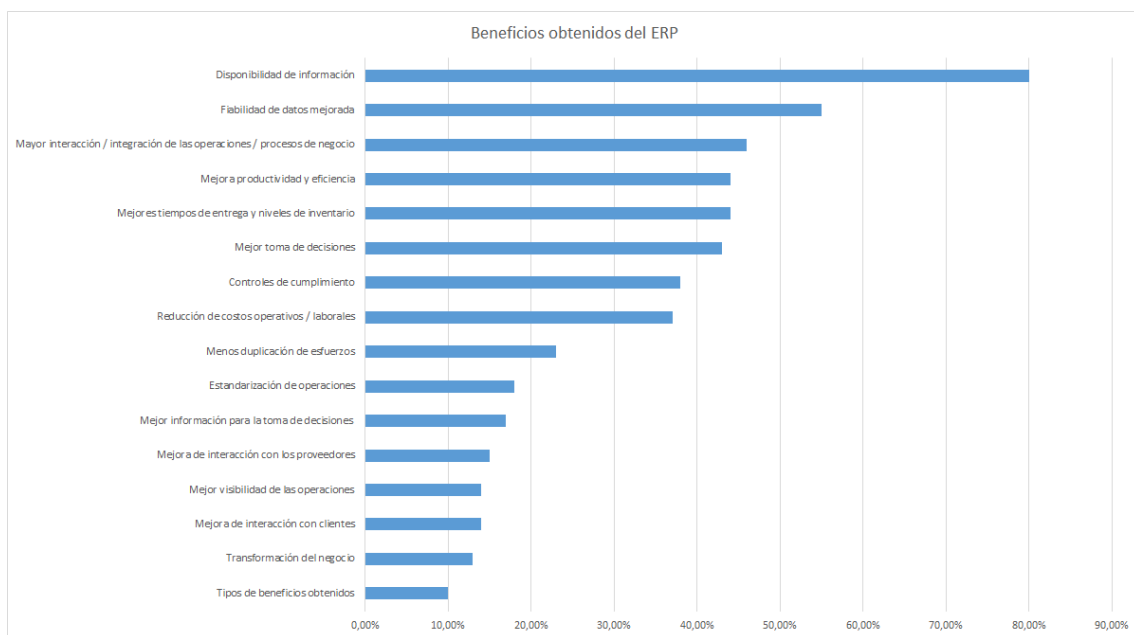


cabo por Panorama Consulting en 2018 [3]. Destacamos que los principales motivos son para mejorar el funcionamiento del negocio, que la compañía crezca y reducir el capital de trabajo.



**Figura 2.1:** Gráfico de los motivos de elección ERP, según una encuesta de Panorama Consulting de 2018

El mismo informe proporciona datos de los beneficios una vez implementado un ERP en una empresa, destacando la disponibilidad de la información y la fiabilidad de los datos. Podemos observar los principales beneficios en la figura 2.2.



**Figura 2.2:** Gráfico de los beneficios de los ERPs, según una encuesta de Panorama Consulting de 2018

## 2.2 Principales ERP

Cómo hemos podido observar los ERPs son útiles para todo tipo de empresas, desde multinacionales a PYMEs, pero debemos tener en cuenta diversos factores a la hora de elegir el ERP adecuado. A grandes rasgos diferenciamos dos tipos de ERPs, los verticales

y los horizontales. Hablamos de verticales cuando el software es especializado para una industria concreta, que proporciona estándares de funcionamiento para dicho sector. En cambio los horizontales sirven para administrar cualquier empresa, permitiendo la personalización para cada caso. Permite que el usuario lo modifique de una forma sencilla y flexible, se trata de un ERP más básico.

Dependiendo del tipo de instalación diferenciamos tres tipos:

- **Instalación en local.** Llamado en inglés *On-premise*, se trata de la instalación clásica de software, en los servidores y equipos de la empresa. Se debe tener en cuenta que se dispone de la infraestructura necesaria, así como las versiones correctas.
- **La nube.** Una empresa externa, normalmente la que desarrolla el software, da soporte lógico y maneja los datos de la empresa cliente. Esta solo necesita acceder a internet para acceder al software. De ahí viene el nombre de la nube, ya que se ejecuta remotamente. Dentro de los servicios en la nube distinguimos tres tipos: PaaS, IaaS y SaaS, pero con *Cloud ERP* nos referimos a las dos primeras ya que se trata simplemente de la instalación tradicional pero en un host fuera de la empresa.
- **Software como Servicio.** De las siglas en inglés SaaS (*Software as a Service*) Es un modelo de distribución de software que cada vez es más utilizado debido a lo práctico que es para el cliente. Cuando nos referimos a la nube se trata de un tipo de arquitectura, sin embargo SaaS es más bien un modelo de monetización de software en la nube. Depende de los intereses hay distintas suscripciones de pago, así como servicios ofrecidos.

Como podemos ver en las figura 2.3 y 2.4 en tan solo un año ha evolucionado mucho el mercado. Los ERP en local tradicionales han pasado de un 67% a un 15%, dejando como soluciones más populares a las opciones en la nube. También podemos observar que la opción de ERP como servicio es la más popular en la actualidad, con un 64%.

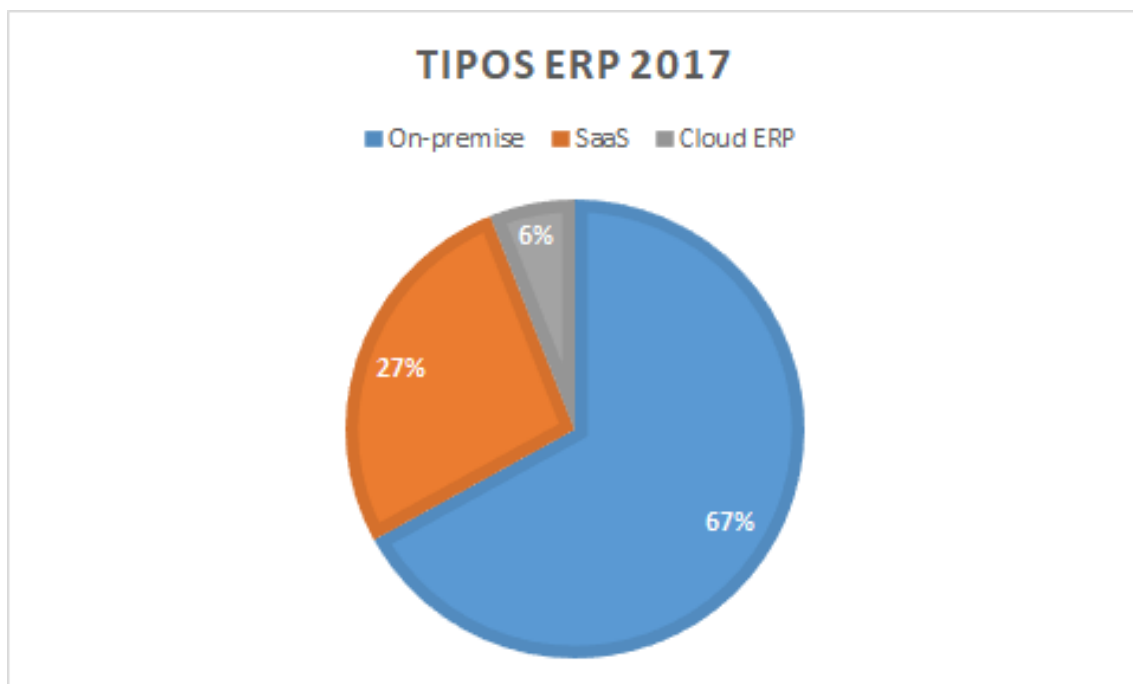


Figura 2.3: Gráfico de los tipos de ERPs en 2017, según una encuesta de Panorama Consulting de 2018

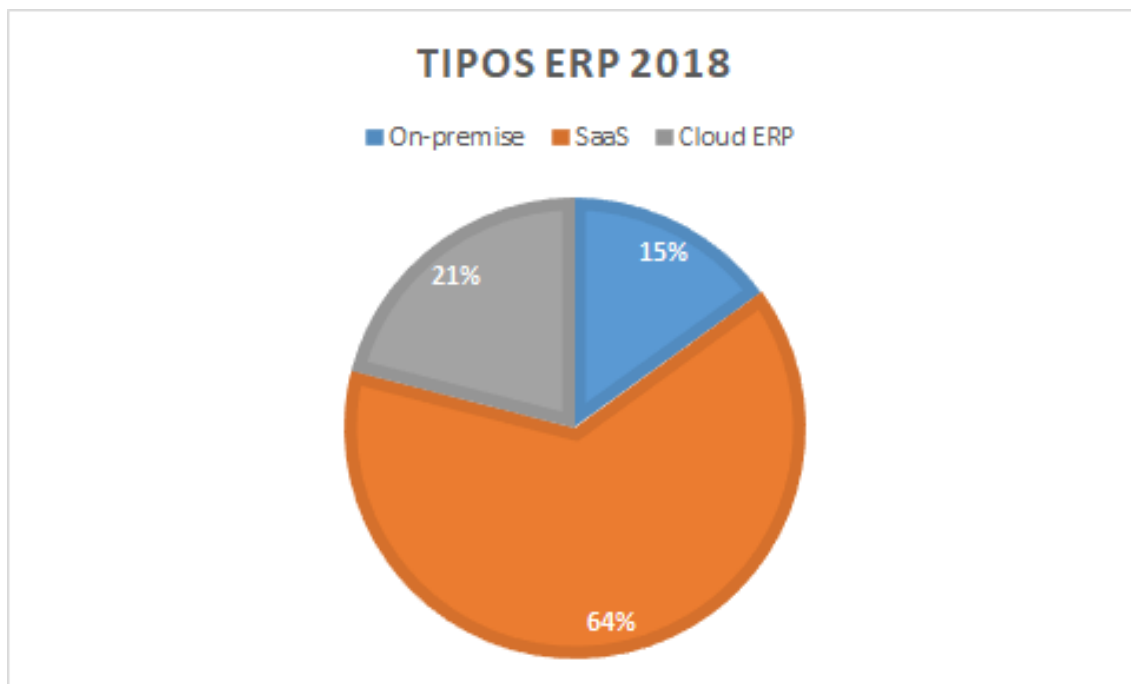


Figura 2.4: Gráfico de los tipos de ERPs en 2018, según una encuesta de Panorama Consulting de 2018

A la hora de elegir una solución ERP o otra debemos tener en cuenta una serie de factores. En primer lugar las necesidades de la empresa, ya que dependiendo del sector o la profundidad de personalización que necesitemos convendrá un tipo de software u otro. Como ya hemos comentado hay ERPs más verticales y otros más horizontales, incluso nos podemos conformar con uno prediseñado. También debemos conocer el volumen de trabajo, ya que limitará los módulos que podamos utilizar. Otro factor que determinará la solución es el presupuesto. Los servicios prediseñados resultan baratos, en cambio, con un mayor presupuesto puedes adquirir soluciones más completas. La empresa interesada debe estudiar primero la rentabilidad del proyecto, para evitar salirse del presupuesto máximo. Por último debemos estudiar la estructura interna de la empresa, sus departamentos y cómo trabajan, para así elegir los módulos que mejor encajen en nuestro negocio.

Un último aspecto diferenciador entre ERP, y software en general, es si se trata de software libre o no. Según *Free Software Foundation*, el software libre es todo programa informático cuyo código fuente puede ser estudiado, modificado, y utilizado libremente con cualquier fin y redistribuido sin o con cambios y/o mejoras. Esto se aplica también al software ERP, que nos da la posibilidad de acceder y adaptar el código fuente, a diferencia del software propietario. Por otra parte, el ERP libre es más barato de implementar, debido a que el coste de licencias es nulo o muy reducido.

A continuación se explican los principales ERP en uso, según los datos de la encuesta que observamos en la figura 2.5, indicando sus características y peculiaridades:

- **Oracle ERP.** Desarrollado por Oracle Corporation, se trata de la opción de software propietario más común. Es un conjunto de aplicaciones software basado en la nube, que fue introducido en 2012, se puede implementar tanto nube pública, privada o híbrida. Se conforma de siete módulos diferentes, y admite funciones empresariales internacionales, como múltiples idiomas y monedas, gestión de sedes, etc.

- **SAP ERP.** Es el segundo ERP propietario más utilizado, esta vez de la compañía alemana SAP SE. Incluye los siguientes procesos de negocio: Operaciones, Finanzas, Administración de Recursos Humanos y Servicios Corporativos. Su última versión es de 2006, sin embargo se actualiza y mejora varias veces al año. Está desarrollado en C, C++ y ABAP, y soporta casi todas las plataformas y sistemas operativos del mercado.
- **Microsoft Dynamics.** Otra opción de software propietario, desarrollada por Microsoft, y forma parte Microsoft Dynamics AX, Microsoft Dynamics GP, Microsoft Dynamics NAV, Microsoft Dynamics SL and Microsoft Dynamics C5. de su software para negocios. Como los anteriores se compone a su vez de otras partes, en este caso las siguientes: Microsoft Dynamics AX, Microsoft Dynamics GP, Microsoft Dynamics NAV, Microsoft Dynamics SL y Microsoft Dynamics C5.
- **Sage.** Basada en Microsoft SQL, está principalmente orientado a pequeños y medianos negocios. Desarrollada por la empresa con el mismo nombre, Sage, desde 2004.
- **Epicor.** Multiplataforma, y disponible tanto en la nube como local. Contiene también un sistema de informes y análisis avanzado. Desarrollado por la empresa Epicor Software Corporation.
- **Infor.** De la mano de la empresa Infor, implementando sus aplicaciones en la nube con la tecnología Amazon Web Services y plataformas de código abierto.
- **IFS.** La empresa sueca IFS AB ofrece otra alternativa de ERP propietario bastante común, con más de 30 años de experiencia en este tipo de software.
- **NetSuite.** Antigua compañía de *cloud computing* estadounidense, que fue comprada por Oracle Corporation en 2016.
- **Odoo.** Una de las opciones de ERP libre más común, y sobre la que se desarrolla este proyecto. Integra muchos módulos que se comunican entre sí de manera eficiente, además se trata de un ERP amigable con una interfaz clara. Se entrará en más detalle en los siguientes apartados.
- **Otras opciones ERP.** Como se puede apreciar en la gráfica 2.5 una gran parte de los ERPs implementados es muy variada, esto incluye Odoo ya comentado, y otros muchos ERPs tanto libres como propietarios. A continuación se mencionan unos cuantos: ADempiere, Apache OFBiz, Dolibarr, ERPNext, Metasfresh, Opentaps, WebERP, abas ERP, ERPNext, SYSPRO, y un largo etcétera.

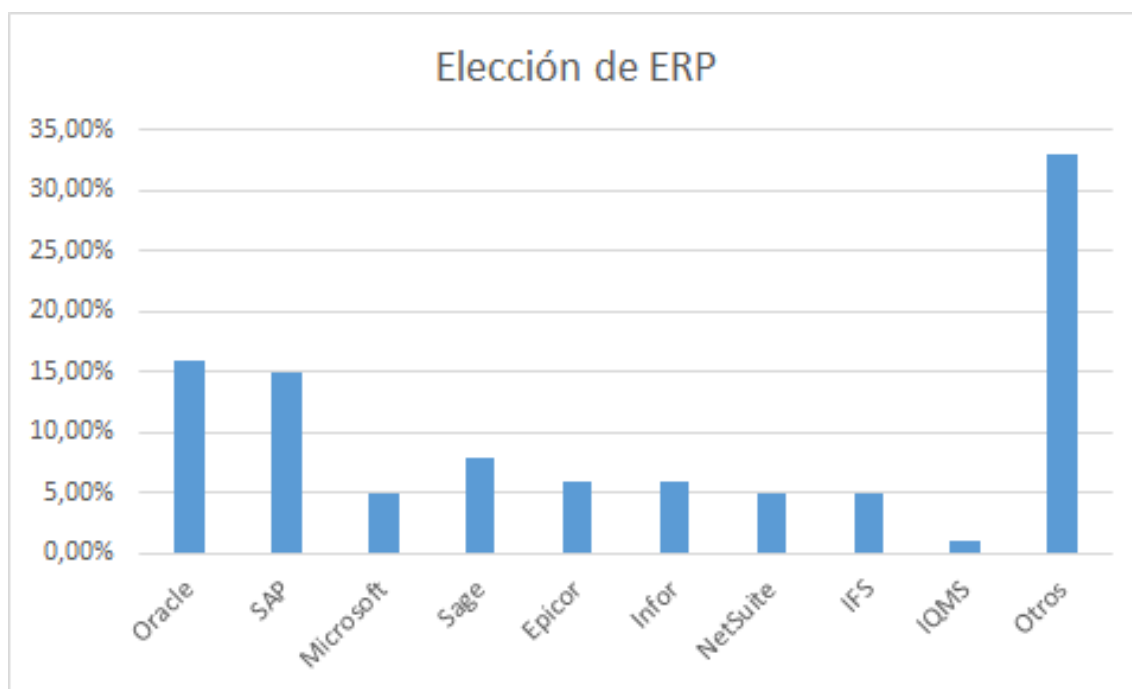


Figura 2.5: Gráfico de la elección de ERPs, según una encuesta de Panorama Consulting de 2018

## 2.3 Elección de Odoo

Entre las principales características de Odoo encontramos que es un ERP libre, multiplataforma y además gratuito, pero eso no significa que no sea completo. De hecho se trata de un ERP muy completo y personalizable, que además es sencillo de utilizar y con una interfaz muy intuitiva.

Antes de empezar con sus características debemos diferenciar sus dos versiones: la *community edition* (CE) y la *enterprise edition* (EE). La CE es la versión software libre, proporciona todas las funciones del *framework* y las funciones esenciales para la mayoría de las aplicaciones empresariales. Tiene una licencia LGPL (*GNU Lesser General Public License*), lo que asegura su libertad para compartir y modificar el software, asegurando que será libre para todos los usuarios. Además esto permite añadir extensiones propietarias sobre los módulos libres. Esta es la versión que utilizaremos, ya que nos sirve para nuestros objetivos. La versión EE está construida encima de la CE, lo que incluye todo lo de la CE además de exclusividades, como por ejemplo interfaz para dispositivos móviles, funcionalidad de reuniones o integración de VoIP. Para más detalles aquí el enlace de la web oficial: <https://www.odoo.com/page/editions>. Cabe destacar que aunque el software es gratuito, implantarlo no, no se pagan licencias de uso pero si su desarrollo y personalización.

Una de las mayores ventajas de Odoo es su libertad. No dependes de ningún proveedor, ya que Odoo no pertenece a ningún distribuidor, puedes elegir el que más convenga a tu proyecto. Al ser filosofía *Open Source* se contrata solo para la parte que se necesita, no hace falta modificar todo el producto solo lo que necesitamos. Aparte podemos modificar el código para mejorar cualquier cosa, o adaptarla a nuestro sistema. Otra propiedad es la conectividad con otros servicios software utilizando *webservices*, se pueden visualizar informes en el estándar PDF, se pueden importar y exportar archivos en formato CSV de hojas de cálculo, o plataformas como Magento y Prestashop.

Odoo es flexible y multiplataforma. Es modular, lo que permite implementar solo las partes que necesitemos, con la posibilidad de ampliación y mejora en un futuro, y siempre compartiendo el mismo flujo de trabajo y base de datos. Su interfaz web le permite ejecutarse en cualquier sistema operativo.

Al ser un ERP extendido y bastante popular cuenta con un gran número de módulos ya creados, lo que nos ahorra tiempo de trabajo si lo que buscamos ya está creado o se puede ampliar algún módulo existente. Esto también se aplica a la comunidad de desarrolladores que hay detrás, con más de 1500 usuarios activos, que siempre puedes encontrar en foros dispuestos a ayudar en cualquier problema. Odoo ofrece 30 aplicaciones, pero además, hay más de 4000 creadas por la comunidad.

Por último comentar la robusta tecnología que utiliza, como es el caso de PostgreSQL, Python y XML. PostgreSQL es uno de los motores de base de datos más utilizados por muchas organizaciones y aplicaciones, además es muy potente y también de desarrollo libre. Python es un lenguaje de programación interpretado muy popular, es multiparadigma, de tipado fuerte, dinámico y multiplataforma. Es un lenguaje bastante simplificado y rápido, pero a su vez elegante y flexible. También tiene una comunidad muy activa, así como multitud de librerías. Finalmente XML (en inglés *eXtensible Markup Language*) es el meta-lenguaje de marcas desarrollado por W3C para almacenaje de datos simple. En el caso de Odoo se utiliza para definir la interfaz, entre otras cosas.

## 2.4 Historia de Odoo

---

En este apartado comentamos brevemente la historia de Odoo, repasando tanto sus versiones como la empresa. Su primera versión llegó al mercado en 2004, desarrollado por la empresa belga OpenERP. La empresa actualmente se llama Odoo S.A., y se fundó en 2002 por Fabien Pinckaers, quién a día de hoy sigue siendo el CEO. Originariamente Odoo se llamaba Tiny ERP, posteriormente en 2009 pasa a llamarse OpenERP. El nombre actual llega en 2014 con la octava versión. Pese a ser código libre la empresa se mantiene y crece debido a que su modelo de negocio está orientado a los servicios, no a licencias de productos. Existen otros productos libres de éxito que siguen el mismo modelo, como es el caso de RedHat Linux, Android OS o Mozilla Firefox. A día de hoy la empresa Odoo S.A. tiene seis oficinas internacionales, actividades en 120 países y más de 550 socios oficiales. La utilizan más de 2.000.000 usuarios, desde empresas muy pequeñas a grandes multinacionales. Entre sus clientes más destacados están: Danone, Auchan (Alcampo), Toyota, Veolia, LaPoste, World Wildlife Fund, Canonical, Singer, Exki, y otras muchas más. Odoo S.A. tiene una red de partners, con cuotas que ayudan a sufragar gastos de desarrollo. Sólo desarrollan el núcleo de la aplicación, de las funcionalidades específicas y las parametrizaciones se encargan los partners.

A continuación en la tabla 2.1 podemos ver las versiones de Odoo [1]. Las versiones 9, 10 y 11 todavía son estables y tienen soporte, las más antiguas no. La actual es la 13, y está prevista la 14 para octubre de 2020.

Nombre del Programa	Versión	Fecha lanzamiento	Cambios Significativos	Tipo licencia de Software
Tiny ERP	1.0	Febrero de 2005	Primera versión	GNU GPL
	2.0	Mayo de 2005		GNU GPL
	3.0	Septiembre de 2005		GNU GPL
	4.0	Diciembre de 2006		GNU GPL
	5.0	Abril de 2009		GNU GPL
OpenERP	6.0	Enero de 2011	Primer cliente web	GNU GPL
	6.1	Febrero de 2012	Primer cliente web Ajax, se descontinúa cliente GTX	GNU GPL
	7.0	22 de diciembre de 2012	Se mejora el cliente web y la usabilidad	GNU GPL
Odoo	8.0	18 de septiembre de 2014	Soporte para CMS: Generador de sitios web, comercio electrónico, puntos de venta e inteligencia de negocio	GNU GPL
	9.0	1 de octubre de 2015	Edición comunitaria	GNU LGPL v3
	9.0	1 de octubre de 2015	Edición Empresarial	Odoo Enterprise Edition License v1.0
	10.0	5 de octubre de 2016	Edición comunitaria	GNU LGPL v3
	10.0	5 de octubre de 2016	Edición Empresarial	Odoo Enterprise Edition License v1.0
	11.0	5 de octubre de 2017	Edición comunitaria	GNU LGPL v3
	11.0	3 de octubre de 2017	Edición Empresarial	Odoo Enterprise Edition License v1.0
	12.0	3 octubre de 2018	Edición comunitaria	GNU LGPL v3
	13.0	5 octubre de 2019	Edición Empresarial	Odoo Enterprise Edition License v1.0
	13.0	5 octubre de 2019	Edición comunitaria	GNU LGPL v3

Tabla 2.1: Historial versiones de Odoo

---

## 2.5 Ediciones de Odoo

---

Como acabamos de ver en la tabla 2.1 existen dos versiones diferentes de Odoo desde el año 2015. Se trata de la Edición Empresarial (Odoo EE, *Odoo Enterprise Edition*) y la edición comunitaria (Odoo CE, *Odoo Community Edition*). En este proyecto se utiliza la edición comunitaria, ya que es totalmente gratuita, entre muchas otras diferencias que se comentan brevemente a continuación:

- En primer lugar el tipo de licencia de software que utilizan ambas versiones. Odoo EE utiliza una licencia privativa <https://www.odoo.com/documentation/user/12.0/legal/terms/enterprise.html>, la cual tenemos que pagar a la empresa para hacer uso de esta versión. En cambio Odoo CE se distribuye bajo la licencia GNU LGPL v3 <https://www.gnu.org/licenses/lgpl-3.0.html>, completamente gratis y *Open Source*. En ambas versiones se nos permite acceder al código fuente, pero en la Edición Empresarial no es recomendado editar este, más bien utilizar complementos de personalización que ofrece la versión.
- La Edición Empresarial incluye más módulos, pero en ambas versiones puedes acceder a la tienda de módulos desarrollados por empresas externas, los cuales pueden ser de pago, gratuitos o de suscripción.
- Encontramos otra diferencia a la hora de actualizar Odoo a una nueva versión. La actualización de Odoo EE corre a cargo de Odoo, y puede haber versiones intermedias antes de la versión final anual. En cambio Odoo no se hace cargo de actualizar la versión Comunitaria, de la cual la propia empresa se tendría que hacer cargo ya sea por ellos mismos o a través de la empresa que implementó el ERP.
- Odoo CE no incluye interfaz móvil, tanto para dispositivos Android como Apple. Ambas versiones incluyen la versión de navegador para ordenador.
- Odoo EE incluye Odoo Studio (<https://www.odoo.com/page/studio>), que añade muchas opciones de personalización.
- Por último comentar que muchos módulos están disponibles en ambas versiones, sin embargo la Edición Empresarial trae más añadidos. Algunos ejemplos son:
  - La gestión de ventas incluye integración VoIP, firma electrónica, servicio de campo, etc.
  - Recursos Humanos incluye nóminas, valoraciones, aprobaciones, recomendaciones y tablero de departamentos.
  - El inventario incluye escáner de barras, conector con servicios de envío (DHL Express, UPS, Fedex, etc.), entre otros extras.
  - El creador de sitios web permite editar formularios, versionado, A/B testing y bloques de llamada a acción.

Para más información la web oficial desglosa todas las diferencias entre las ediciones: ([https://www.odoo.com/es\\_ES/page/editions](https://www.odoo.com/es_ES/page/editions)).

---

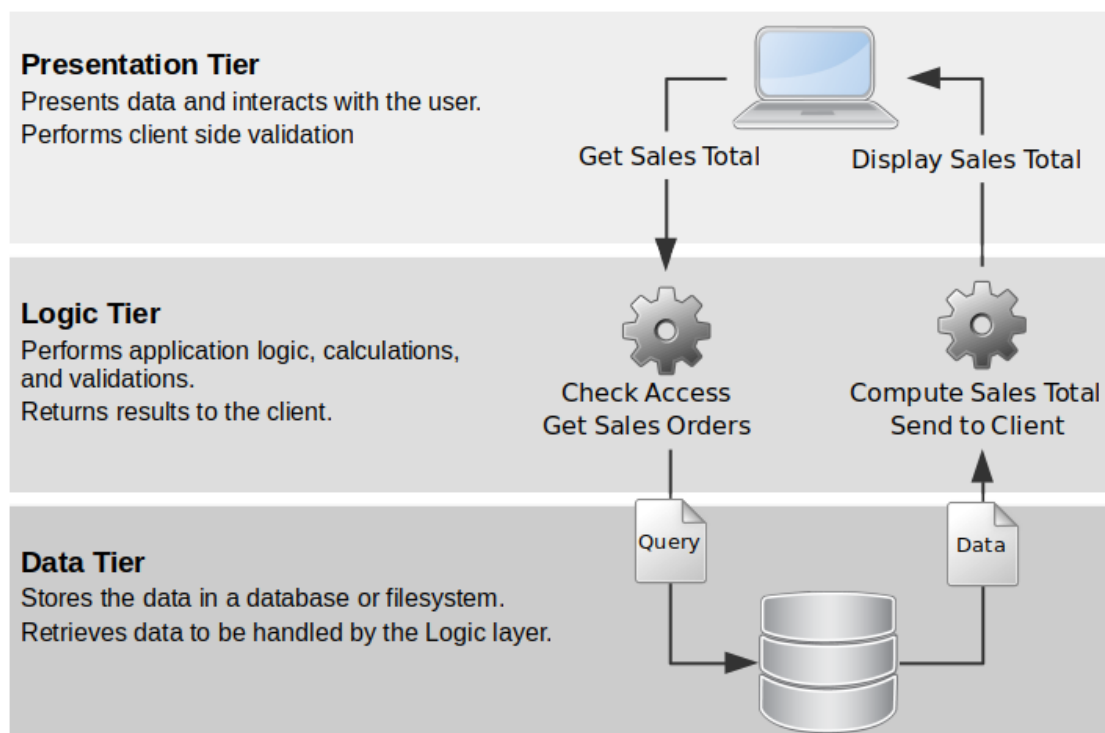
## 2.6 Arquitectura

---

A continuación se explican las distintas partes que componen la arquitectura de Odoo[2]:



- **Capa de datos.** Representa la capa más baja, almacena y da persistencia a los datos mediante la tecnología PostgreSQL. Otros archivos binarios como adjuntos, documentos o imágenes, se almacenan en los directorios del sistema. Debemos tener en cuenta lo anterior a la hora de realizar copias de seguridad. La capa de datos proporciona información a la capa superior, la lógica.
- **Capa Lógica.** Es la capa que contiene los programas que se ejecutan, también reciben y envían peticiones del usuario, así como solicitar datos a la base de datos. Es la capa que comunica la inferior con la superior, los datos con la interfaz que ve el usuario.
- **Capa de presentación.** Por último la capa de más alto nivel es la interfaz gráfica que ve el usuario por pantalla. Es intuitiva y sencilla, y se comunica únicamente con la capa lógica de forma ágil.



**Figura 2.6:** Figura de la arquitectura básica de Odoo del libro *Odoo 12 Development Essentials*, Daniel Reis

Como podemos ver sigue el patrón de arquitectura software Modelo Vista Controlador (MVC), modelo multicapa utilizado por casi todas las aplicaciones informáticas. Nos podemos referir a la capa de datos como modelo o como capa de negocio. La capa de presentación también se denomina vista o interfaz de usuario. Por último la capa lógica también se llama controlador. Estos términos son sinónimos, y aparecen a lo largo de la memoria.



---

---

## CAPÍTULO 3

# Análisis

---

Una vez contextualizada la tecnología procedemos a presentar y analizar en profundidad el problema. En el capítulo 3: Análisis empieza con una especificación de requisitos siguiendo el estándar IEEE 830 y termina con un ejemplo de uso de la aplicación concreto.

### 3.1 Especificación de requisitos software

---

#### 3.1.1. Introducción

En este apartado se redacta una Especificación de requisitos software (ERS) sobre el caso propuesto, siguiendo el estándar IEEE 830 [4]. En la introducción se abordan los siguientes puntos: propósito, ámbito del sistema, definiciones, siglas, abreviaturas, referencias y visión global.

##### Propósito

La meta del ERS es definir la idea a desarrollar y sus funciones básicas, proporcionando así al desarrollador una descripción del problema que le facilite su diseño e implementación. Así mismo el documento también debe ser revisado por el cliente, para comprobar que el documento ERS encaja con el producto que él encarga.

##### Ámbito del sistema

Como ya se ha comentado, al producto se le llamará SportService, y se encargará de gestionar los servicios deportivos, con sus respectivos clientes y personal. Se puede dar de alta, de baja y editar tanto clientes como trabajadores de la empresa (por ejemplo monitores de una actividad deportiva). Se podrán crear, borrar y editar servicios, así como inscribir y desinscribir clientes a estos. Consultar si tiene acceso y notificaciones. El sistema permite otras opciones de gestión, como comprobar accesos y realizar notificaciones.

##### Definiciones, acrónimos y abreviaturas

A continuación se definen distintos términos utilizados en el apartado ERS:

- **ERP:** Explicado detalladamente en la sección 2.1.
- **Framework:** Proviene del inglés, llamado también entorno de trabajo o marco de trabajo. Es una estructura conceptual y tecnológica de asistencia definida, normal-

mente, con artefactos o módulos concretos de software, que puede servir de base para la organización y desarrollo de software [1].

- **PostgreSQL:** Sistema de gestión de base de datos relacional, abierto y basado en el lenguaje SQL.
- **XML:** Las siglas en inglés son: *eXtensible Markup Language*, lenguaje de marcado extensible o lenguaje de marcas extensible. Se trata de un metalenguaje que se utiliza para almacenar datos de forma legible, para bases de datos, comunicación entre aplicaciones o integración de datos.
- **Python:** Lenguaje de programación interpretado, multiparadigma, con soporte para la programación orientada a objetos, programación imperativa y funcional. Se caracteriza por tener una sintaxis legible y ser un lenguaje dinámico y multiplataforma.
- **MVC:** Se trata de un popular patrón de arquitectura software que separa datos y lógica de negocio, separando en tres componentes: Modelo, Vista y Controlador.

### Visión general del ERS

El siguiente apartado se trata de una descripción general del producto, para contextualizarlo y facilitar la comprensión de los requisitos, los cuales se enumeran en el último apartado. A su vez la descripción general consta de una serie de subsecciones: perspectiva del producto, funciones del producto, características de los usuarios, restricciones, suposiciones, dependencias y futuros requisitos. Por último el apartado de requisitos específicos, divididos en requisitos de interfaz, requisitos funcionales y otros requisitos.

#### 3.1.2. Descripción general

##### Perspectiva del Producto

El producto a desarrollar no depende de ningún otro software, ni forma parte de un sistema mayor. Al tratarse de una aplicación Odoos se debe instalar en nuestro sistema operativo, así como instalar y configurar PostgreSQL y disponer de un navegador web. La instalación se explica detalladamente en el apéndice A, y posteriormente la implantación en el apartado 6.

##### Funciones del producto

El objetivo de la aplicación es gestionar los clientes de la empresa, creándolos o inscribiéndolos a los cursos o servicios ofrecidos. Las funciones a concretar son:

- Dar de alta, de baja y editar clientes
- Dar de alta, de baja y editar trabajadores
- Crear, borrar y editar servicios, así como inscribir y desinscribir clientes a estos.
- Consulta de accesos.
- Notificación de inscripción.

## Características de los Usuarios

El uso de este software va dirigido a dos tipos de usuarios diferentes: al trabajador de la empresa y al administrador.

El trabajador será el usuario básico de la aplicación, con la que podrá desempeñar todas las funcionalidades que hemos comentado anteriormente. No necesita conocimientos informáticos, ya que para el uso de este producto solo requiere un mínimo de conocimientos de navegación web básicos. Una vez implantado el software en su puesto de trabajo, se le formará con las instrucciones necesarias para su uso diario, arrancar el sistema Odoo y uso de la aplicación concreta.

En cambio el administrador no utilizará normalmente la aplicación, ya que su usuario solo sirve para realizar cambios en el ERP. Él tiene los permisos necesarios para añadir o modificar módulos, añadir nuevos cursos, eliminar cursos actuales, y demás funciones que **NO están definidas** en el alcance de la aplicación original. Este usuario es el desarrollador de Odoo, por lo tanto debe tener conocimientos medios/avanzados en informática, para manejarse con las tecnologías necesarias para desarrollar en Odoo, comentadas en el apartado 2.6.

## Restricciones

Para que la aplicación funcione nuestra computadora tiene que cumplir una serie de condiciones:

- Máquina (tanto física como virtual) con sistema operativo Ubuntu 18.04 o superior.
- Navegador web actualizado (Google Chrome, Firefox, por ejemplo).
- PostgreSQL 9.6 instalado en el sistema.
- Python 3.7 con las librerías necesarias para las dependencias de Odoo instalado.
- Correcta instalación de Odoo 12 en el sistema.

Todo lo comentado anteriormente se explica brevemente su instalación y configuración en el Anexo A.

## Suposiciones y dependencias

Relacionado con el apartado anterior, el desarrollo se ha realizado sobre las versiones actuales más estables, por tanto cualquier actualización futura puede dar problemas en alguna funcionalidad del sistema. Es decir, cualquier actualización de Ubuntu, PostgreSQL, Python o Odoo puede afectar. Se tendrían que revisar las notas de actualización, con el objetivo de identificar cambios que afecten al software desarrollado. Dado el caso se podría modificar lo que sea necesario para encajar con los cambios de las actualizaciones. Los cambios pueden ser sencillos, como una simple actualización de una librería de Python, o complejos, como una actualización de Odoo que cambia radicalmente alguna función utilizada en muchas partes del desarrollo.

## Requisitos futuros

Una vez realizada la aplicación básica para la gestión de los servicios se podrían incorporar muchas funcionalidades, todo depende de los recursos dispuestos a invertir en

el proyecto. La primera funcionalidad que se debería incorporar es un sistema de pagos para los clientes. Otra mejora interesante sería un portal para acceder los clientes, enlazado al sistema Odoo. Desde este portal pueden gestionar su propio usuario, así como inscribirse o desinscribirse en distintos servicios. También mediante este portal se pueden gestionar pagos integrados, notificaciones varias al usuario, entre otros extras. En el apartado 8 se profundiza en este tema.

### 3.1.3. Requisitos específicos

En este apartado se plantean los requisitos detallados para que los diseñadores trabajen sobre ellos, y posteriormente se puedan realizar pruebas y verificar su cumplimiento. Nos centramos en los requisitos funcionales. Los requisitos específicos se muestran en forma de tabla, con el siguiente contenido:

- **Identificación de requerimiento:** Código único de referencia para cada requisito.
- **Nombre del requisito:** Nombre que se le da al requisito concreto.
- **Descripción del requisito:** Breve explicación del requisito.
- **Entrada:** Situación en la que nos encontramos antes del requisito. Condiciones a cumplir.
- **Proceso:** Acciones automáticas que realiza el sistema para dicho requisito.
- **Salida:** Situación en la que nos quedamos una vez se procesa el requisito.

<b>Código de identificación</b>	RE01
<b>Nombre de requisito</b>	Inicio de sesión a la aplicación
<b>Descripción</b>	El sistema debe ofrecer una inicio de sesión al usuario, que le permita utilizar la aplicación.
<b>Entrada</b>	Odoo debe estar iniciado para poder iniciar sesión. Se introducen los credenciales de usuario (nombre y contraseña).
<b>Proceso</b>	El sistema comprueba que los datos introducidos son correctos.
<b>Salida</b>	En caso de éxito, se puede utilizar la aplicación al completo una vez iniciada la sesión. En caso contrario, aparece un mensaje de error.

Tabla 3.1: Requisito específico RE01

<b>Código de identificación</b>	RE02
<b>Nombre de requisito</b>	Listar clientes
<b>Descripción</b>	Obtenemos una lista con clientes de la empresa.
<b>Entrada</b>	Debemos iniciar sesión para listar los clientes. El usuario puede manejar una serie de filtros de búsqueda para acotar la lista.
<b>Proceso</b>	El sistema realiza una consulta en la base de datos según los filtros de búsqueda introducidos por el usuario.
<b>Salida</b>	Por defecto vemos todos los clientes de la empresa, y según los filtros de búsqueda podemos ver clientes más concretos.

**Tabla 3.2:** Requisito específico RE02

<b>Código de identificación</b>	RE03
<b>Nombre de requisito</b>	Añadir un cliente
<b>Descripción</b>	Añadimos un nuevo cliente al sistema.
<b>Entrada</b>	Debemos iniciar sesión para crear nuevos clientes. Hay que introducir unos datos mínimos para su creación, es posible añadir otros opcionales.
<b>Proceso</b>	El sistema inserta en la base de datos el nuevo cliente con sus respectivos datos.
<b>Salida</b>	Si no hay ningún error el cliente se añade a la base de datos del sistema. En ambos casos se notifica al usuario con un mensaje.

**Tabla 3.3:** Requisito específico RE03

<b>Código de identificación</b>	RE04
<b>Nombre de requisito</b>	Ver detalles cliente
<b>Descripción</b>	Nos muestra información detallada de un cliente en concreto.
<b>Entrada</b>	Debemos iniciar sesión y listar los clientes, con filtros de búsqueda o no. En el listado de clientes debemos seleccionar un cliente en concreto.
<b>Proceso</b>	El sistema busca el cliente en concreto a la base de datos y obtiene todos sus datos.
<b>Salida</b>	Vemos una ventana con todos los detalles del cliente seleccionado anteriormente.

**Tabla 3.4:** Requisito específico RE04

<b>Código de identificación</b>	RE05
<b>Nombre de requisito</b>	Editar un cliente
<b>Descripción</b>	Nos permite editar los datos de un cliente, así como eliminarlo.
<b>Entrada</b>	Debemos iniciar sesión y listar los clientes, con filtros de búsqueda o no. En el listado de clientes debemos seleccionar un cliente en concreto. Una vez seleccionado tenemos la opción de editar este mismo, actualizando los datos deseados o eliminando el cliente.
<b>Proceso</b>	El sistema procesa los cambios introducidos para actualizar la base de datos en caso de ser posible. Si se elimina un cliente se debe de reflejar en sus respectivos servicios, si esta activo en alguno.
<b>Salida</b>	Volvemos a ver la ventana con todos los detalles del cliente, pero esta vez actualizados. Si hay algún error se notifica mediante un mensaje y no se actualizan los datos.

**Tabla 3.5:** Requisito específico RE05

<b>Código de identificación</b>	RE06
<b>Nombre de requisito</b>	Listar trabajadores
<b>Descripción</b>	Obtenemos una lista con trabajadores de la empresa.
<b>Entrada</b>	Debemos iniciar sesión para listar los trabajadores. El usuario puede manejar una serie de filtros de búsqueda para acotar la lista.
<b>Proceso</b>	El sistema realiza una consulta en la base de datos según los filtros de búsqueda introducidos por el usuario.
<b>Salida</b>	Por defecto vemos todos los trabajadores de la empresa, y según los filtros de búsqueda podemos ver trabajadores más concretos.

**Tabla 3.6:** Requisito específico RE06



<b>Código de identificación</b>	RE07
<b>Nombre de requisito</b>	Añadir un trabajador
<b>Descripción</b>	Añadimos un nuevo trabajador al sistema.
<b>Entrada</b>	Debemos iniciar sesión para crear nuevos trabajadores. Hay que introducir unos datos mínimos para su creación, es posible añadir otros opcionales.
<b>Proceso</b>	El sistema inserta en la base de datos el nuevo trabajador con sus respectivos datos.
<b>Salida</b>	Si no hay ningún error el trabajador se añade a la base de datos del sistema. En ambos casos se notifica al usuario con un mensaje.

**Tabla 3.7:** Requisito específico RE07

<b>Código de identificación</b>	RE08
<b>Nombre de requisito</b>	Ver detalles trabajador
<b>Descripción</b>	Nos muestra información detallada de un trabajador en concreto.
<b>Entrada</b>	Debemos iniciar sesión y listar los trabajadores, con filtros de búsqueda o no. En el listado de trabajadores debemos seleccionar un trabajador en concreto.
<b>Proceso</b>	El sistema busca el trabajador en concreto a la base de datos y obtiene todos sus datos.
<b>Salida</b>	Vemos una ventana con todos los detalles del trabajador seleccionado anteriormente.

**Tabla 3.8:** Requisito específico RE08

<b>Código de identificación</b>	RE09
<b>Nombre de requisito</b>	Editar un trabajador
<b>Descripción</b>	Nos permite editar los datos de un trabajador, así como eliminarlo.
<b>Entrada</b>	Debemos iniciar sesión y listar los trabajadores, con filtros de búsqueda o no. En el listado de trabajadores debemos seleccionar un trabajador en concreto. Una vez seleccionado tenemos la opción de editar este mismo, actualizando los datos deseados o eliminando el trabajador.
<b>Proceso</b>	El sistema procesa los cambios introducidos para actualizar la base de datos en caso de ser posible. Si se elimina un trabajador se debe de reflejar en sus respectivos servicios, si esta activo en alguno.
<b>Salida</b>	Volvemos a ver la ventana con todos los detalles del trabajador, pero esta vez actualizados. Si hay algún error se notifica mediante un mensaje y no se actualizan los datos.

**Tabla 3.9:** Requisito específico RE09

<b>Código de identificación</b>	RE10
<b>Nombre de requisito</b>	Listar servicios
<b>Descripción</b>	Obtenemos una lista con servicios de la empresa.
<b>Entrada</b>	Debemos iniciar sesión para listar los servicios. El usuario puede manejar una serie de filtros de búsqueda para acotar la lista.
<b>Proceso</b>	El sistema realiza una consulta en la base de datos según los filtros de búsqueda introducidos por el usuario.
<b>Salida</b>	Por defecto vemos todos los servicios de la empresa, y según los filtros de búsqueda podemos ver servicios más concretos.

**Tabla 3.10:** Requisito específico RE10

<b>Código de identificación</b>	RE11
<b>Nombre de requisito</b>	Añadir un servicio
<b>Descripción</b>	Añadimos un nuevo servicio al sistema.
<b>Entrada</b>	Debemos iniciar sesión para crear nuevos servicios. Hay que introducir unos datos mínimos para su creación, es posible añadir otros opcionales. Podemos también asignar trabajadores a este servicio, como por ejemplo un monitor para cierta actividad deportiva. Además podemos añadir clientes a este servicio.
<b>Proceso</b>	El sistema inserta en la base de datos el nuevo servicio. Al añadir tanto algún trabajador como algún cliente no pueden solapar este servicio con otros que coincidan en horario. Si se añade algún cliente o se asigna algún trabajador se debe de reflejar en sus respectivos datos.
<b>Salida</b>	Si no hay ningún error el servicio se añade a la base de datos del sistema. En ambos casos se notifica al usuario con un mensaje.

**Tabla 3.11:** Requisito específico RE11

<b>Código de identificación</b>	RE12
<b>Nombre de requisito</b>	Ver detalles servicio
<b>Descripción</b>	Nos muestra información detallada de un servicio en concreto.
<b>Entrada</b>	Debemos iniciar sesión y listar los servicios, con filtros de búsqueda o no. En el listado de servicios debemos seleccionar un servicio en concreto.
<b>Proceso</b>	El sistema busca el servicio en concreto a la base de datos y obtiene todos sus datos.
<b>Salida</b>	Vemos una ventana con todos los detalles del servicio seleccionado anteriormente.

**Tabla 3.12:** Requisito específico RE12

<b>Código de identificación</b>	RE13
<b>Nombre de requisito</b>	Editar un servicio
<b>Descripción</b>	Nos permite editar los datos de un servicio, así como eliminarlo.
<b>Entrada</b>	Debemos iniciar sesión y listar los servicios, con filtros de búsqueda o no. En el listado de servicios debemos seleccionar un servicio en concreto. Una vez seleccionado tenemos la opción de editar este mismo, actualizando los datos deseados o eliminando el servicio. Podemos también asignar trabajadores a este servicio, como por ejemplo un monitor para cierta actividad deportiva. Además podemos añadir clientes a este servicio.
<b>Proceso</b>	El sistema procesa los cambios introducidos para actualizar la base de datos en caso de ser posible. Si se elimina un servicio se debe reflejar desapareciendo en los datos de clientes y/o trabajadores previamente enlazados a este servicio. Al añadir tanto algún trabajador como algún cliente no pueden solapar este servicio con otros que coincidan en horario. Si se añade algún cliente o se asigna algún trabajador se debe de reflejar en sus respectivos datos.
<b>Salida</b>	Volvemos a ver la ventana con todos los detalles del servicio, pero esta vez actualizados. Si hay algún error se notifica mediante un mensaje y no se actualizan los datos.

**Tabla 3.13:** Requisito específico RE13

<b>Código de identificación</b>	RE14
<b>Nombre de requisito</b>	Notificación inscripción
<b>Descripción</b>	El cliente recibe un correo electrónico cuando se inscribe a un servicio, con información del mismo.
<b>Entrada</b>	Debemos iniciar sesión y seleccionar un servicio a editar o crear uno nuevo, al que añadiremos el cliente.
<b>Proceso</b>	Tras el proceso de edición o creación de un servicio el sistema manda un correo electrónico informativo, obteniendo la dirección de los datos del cliente.
<b>Salida</b>	Si hay algún error se notifica mediante un mensaje y no se actualizan los datos, en caso contrario aparece un mensaje de éxito. El cliente recibe un correo electrónico con los datos del servicio.

**Tabla 3.14:** Requisito específico RE14

<b>Código de identificación</b>	RE15
<b>Nombre de requisito</b>	Notificación baja
<b>Descripción</b>	El cliente recibe un correo electrónico informativo cuando se elimina de un servicio inscrito o el propio servicio desaparece.
<b>Entrada</b>	Debemos iniciar sesión y seleccionar un servicio a editar, al que eliminaremos el cliente o el servicio.
<b>Proceso</b>	Tras el proceso de edición del servicio el sistema manda el correo electrónico informativo, obteniendo la dirección de los datos del cliente.
<b>Salida</b>	Si hay algún error se notifica mediante un mensaje y no se actualizan los datos, en caso contrario aparece un mensaje de éxito. El cliente recibe un correo electrónico con los datos del servicio.

**Tabla 3.15:** Requisito específico RE15

<b>Código de identificación</b>	RE16
<b>Nombre de requisito</b>	Cierre de sesión de la aplicación
<b>Descripción</b>	Una vez terminado el uso de la aplicación el usuario cierra su sesión.
<b>Entrada</b>	El usuario debe haber iniciado sesión correctamente.
<b>Proceso</b>	El sistema cierra la sesión actual iniciada con dicho usuario.
<b>Salida</b>	Quedamos en el menú de aplicaciones de Odoo.

**Tabla 3.16:** Requisito específico RE16

## 3.2 Ejemplo de uso concreto

---

Se desea implementar SportService a una empresa ficticia que ofrece numerosos servicios deportivos. La empresa ofrece sus servicios en un gran complejo deportivo a las afueras de la ciudad, con diferentes salas y espacios, con sus respectivos horarios y actividades. Los usuarios de la aplicación son el personal de administración de la empresa. Se necesita una aplicación informática que gestione todo lo que ofrece la empresa, de la forma más automática posible. A continuación una enumeración de servicios y funcionalidades que necesita dicha empresa.

1. Almacenar los distintos clientes de la empresa, dar de alta, modificar y eliminar. Estos son algunos de los datos que se desean guardar:
  - a) Datos personales (DNI, fecha de nacimiento, sexo, dirección, teléfono).
  - b) Identificador único por cliente.
  - c) Correo electrónico.

2. Almacenar los trabajadores de la empresa, dar de alta, modificar y eliminar. Se trata de los monitores de las actividades deportivas que ofrece la empresa. Estos son algunos de los datos que se desean guardar:
  - a) Datos personales (DNI, fecha de nacimiento, sexo, dirección, teléfono).
  - b) Identificador único por trabajador.
  - c) Número de seguridad social.
3. Servicio de fisio mediante citas. De 10:00 a 20:00, los clientes han de reservar con 7 días de antelación. Si está disponible se le acepta la cita, se le notificará 1 día antes mediante email. Puede cancelar 5 días antes de la cita.
4. Quiromasaje De 10:00 a 20:00, los clientes han de reservar con 7 días de antelación. Si está disponible se le acepta la cita, se le notificará 1 día antes mediante email. Puede cancelar 5 días antes de la cita.
5. Sesiones de rayos uva. Horario de 15:00 a 21:00, aforo máximo de 5 personas en sesiones de 30 minutos. Se puede reservar con 2 días o menos de antelación, o acudir el mismo día.
6. Acceso a sauna. Horario de 9:00 a 21:00, aforo máximo de 5 personas en sesiones de 30 minutos. Se puede reservar con 2 días o menos de antelación, o acudir el mismo día.
7. Suscripción gimnasio. Acceso de 6:00 a 00:00. Los clientes pueden matricularse 1, 3 o 12 meses. Durante esos periodos tienen libre acceso a las salas de gimnasio. Al finalizar la suscripción se les notifica mediante email, con opción de renovar o cancelar (por defecto sigue matriculado).
8. Política de matriculación a cursos:
  - a) Se puede matricular antes de que empiece.
  - b) Trimestral
  - c) Al finalizar el curso se les notifica mediante email.
  - d) Las clases son de 1 hora de duración.
  - e) No se puede matricular si no hay plazas.
  - f) No se puede matricular a cursos que coinciden en horario.
9. Piscina con aforo de 50 personas, 2 cursos de 26 plazas cada uno, también opción de baño libre. Pago por acceder 1 día o suscripción mensual baño libre (acceso todos los días).
10. 2 pistas de pádel, 2 cursos con 4 plazas cada uno. Opción de acceso libre. Pago por acceder 1 día o suscripción mensual acceso pádel libre (acceso todos los días).
11. Pista de tenis con 2 cursos de 4 plazas, acceso libre cuando no se imparte curso. Se puede acceder si tienes reservado o está libre la pista, se puede reservar la pista 2 días antes.
12. Zona de artes marciales, donde se imparten cursos de las más populares, 20 plazas por curso.

Finalmente podemos ver los horarios que ofrece la empresa en las siguientes imágenes 3.1, 3.2, 3.3 y 3.4.

Natación												
	LUNES		MARTES		MIÉRCOLES		JUEVES		VIERNES		SÁBADO	
8:00												
9:00												
10:00												
11:00												
12:00												
13:00												
14:00												
15:00												
16:00												
17:00												
18:00	Blue		Orange		Blue		Orange		Blue		Orange	
19:00	Blue		Orange		Blue		Orange		Blue		Orange	
20:00	Blue		Orange		Blue		Orange		Blue		Orange	
21:00	Blue		Orange		Blue		Orange		Blue		Orange	
22:00	Blue		Orange		Blue		Orange		Blue		Orange	

Figura 3.1: Horario cursos de natación

Pádel												
	LUNES		MARTES		MIÉRCOLES		JUEVES		VIERNES		SÁBADO	
8:00												
9:00												
10:00												
11:00												
12:00												
13:00												
14:00												
15:00												
16:00												
17:00												
18:00	Blue		Orange		Blue		Orange		Blue		Orange	
19:00	Blue		Orange		Blue		Orange		Blue		Orange	
20:00	Blue		Orange		Blue		Orange		Blue		Orange	
21:00	Blue		Orange		Blue		Orange		Blue		Orange	
22:00	Blue		Orange		Blue		Orange		Blue		Orange	
	Pista 1	Pista 2	Pista 1	Pista 2	Pista 1	Pista 2	Pista 1	Pista 2	Pista 1	Pista 2	Pista 1	Pista 2

Figura 3.2: Horario cursos de pádel

Tenis												
	LUNES		MARTES		MIÉRCOLES		JUEVES		VIERNES		SÁBADO	
8:00												
9:00												
10:00												
11:00												
12:00												
13:00												
14:00												
15:00												
16:00												
17:00												
18:00	Blue		Orange		Blue		Orange		Blue		Orange	
19:00	Blue		Orange		Blue		Orange		Blue		Orange	
20:00	Blue		Orange		Blue		Orange		Blue		Orange	
21:00	Blue		Orange		Blue		Orange		Blue		Orange	
22:00	Blue		Orange		Blue		Orange		Blue		Orange	
	Pista 1	Pista 2	Pista 1	Pista 2	Pista 1	Pista 2	Pista 1	Pista 2	Pista 1	Pista 2	Pista 1	Pista 2

Figura 3.3: Horario cursos de tenis

Artes marciales					
	LUNES	MARTES	MIERCOLES	JUEVES	VIERNES
17:00	Judo		Judo		Judo
18:00	Karate	Tai Chi	Karat	Tai Chi	Karate
19:00	Takewondo	Kendo	Takewondo	Kendo	Takewondo
20:00	Kick Boxing	Kung Fu	Kick Boxing	Kung Fu	Kick Boxing
21:00	Boxeo		Boxeo		Boxeo

**Figura 3.4:** Horario cursos de artes marciales



---

---

## CAPÍTULO 4

# Diseño de la solución

---

Tras realizar el análisis del problema, tenemos claro qué vamos a desarrollar, y que requisitos debe cumplir nuestra aplicación. Antes de entrar directamente a desarrollar en Odoó debemos pasar por la etapa de desarrollo que incluye las siguientes tres partes: Diagrama de casos de uso, Diagrama de clases y Prototipado. Cabe destacar que cualquier diseño de este apartado puede cambiar en el resultado final, debido tanto a complicaciones como mejoras a la hora de implementar las ideas.

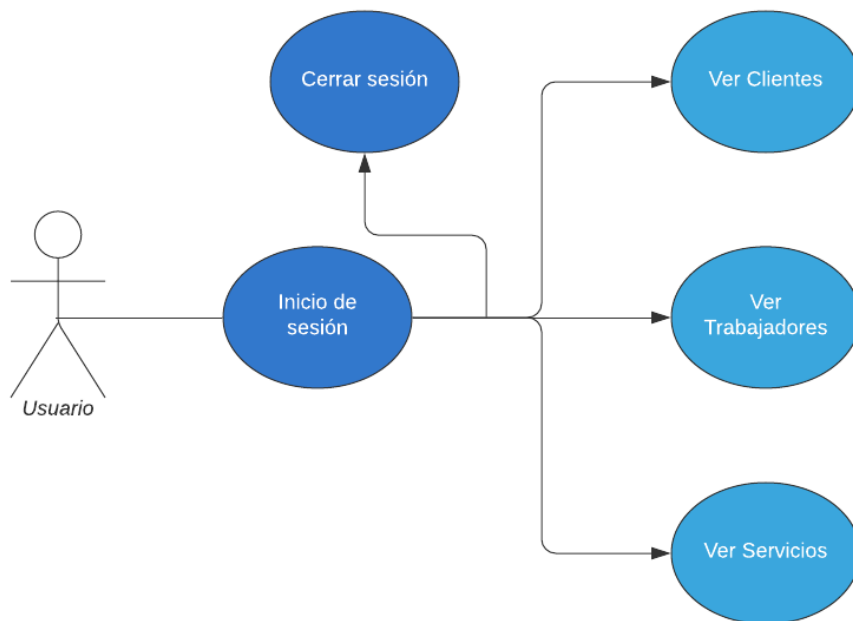
### 4.1 Diagramas de casos de uso

---

Los diagramas de casos de uso sirven para mostrar gráficamente la especificación de requisitos, como se comunica y interacciona nuestro sistema con sus actores. En este caso solo hay un actor: el usuario administrativo que utiliza la aplicación. Los diagramas nos dan un buen punto de vista sobre los actores que intervienen y los procesos.

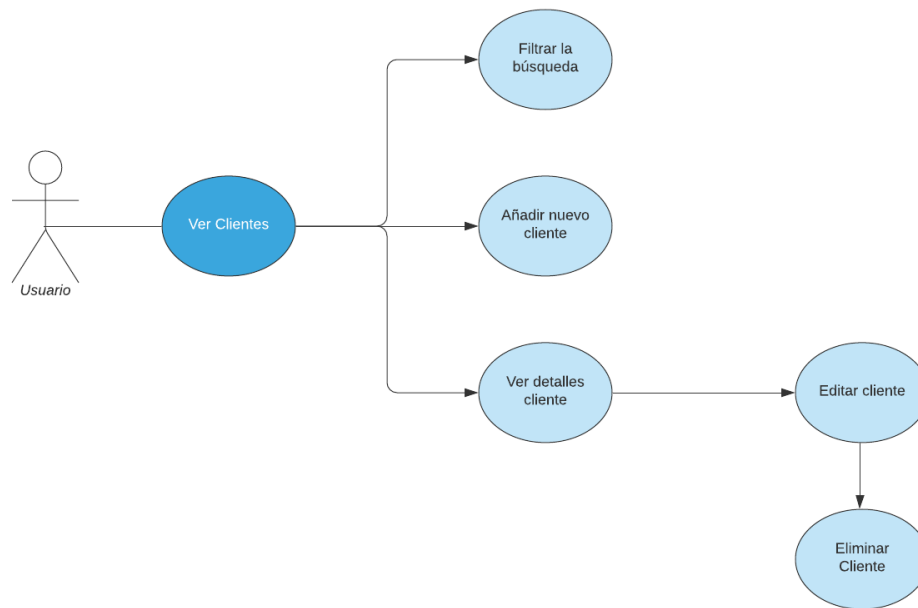
En este apartado se siguen los estándares del lenguaje unificado de modelado (UML, en inglés: *Unified Modeling Language*). Se trata del lenguaje de modelado de sistemas software más popular en la actualidad.

En la figura 4.1 podemos observar un diagrama de caso de uso general de la aplicación. Para poder utilizarla el usuario debe iniciar sesión, posteriormente puede cerrar esta sesión. Tras autenticar puede ver los clientes, trabajadores o servicios que hay en el sistema. En este diagrama se ven reflejados los requisitos específicos: 3.1, 3.16, 3.2, 3.6 y 3.10.



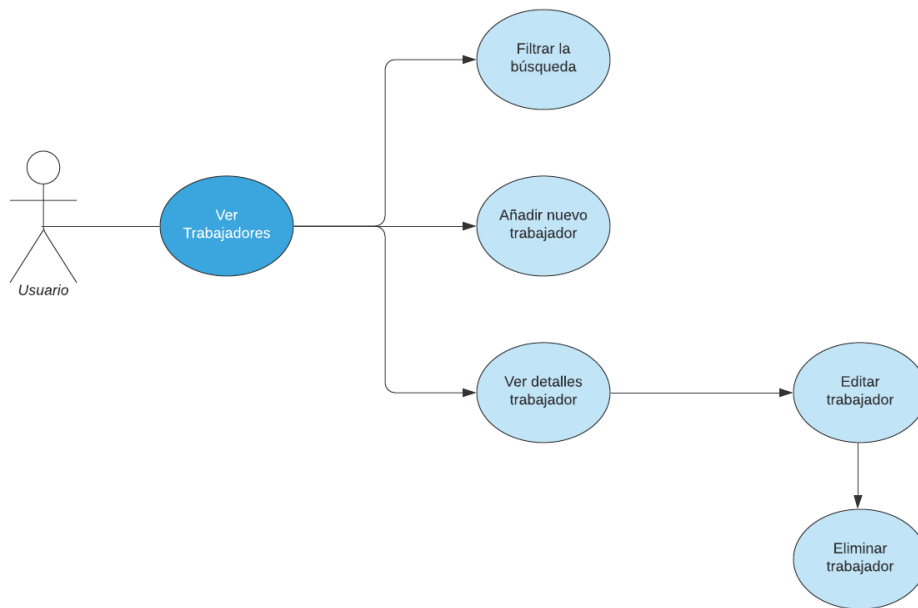
**Figura 4.1:** Diagrama de casos de uso 1

Si elegimos ver clientes el diagrama sigue según la figura 4.2. Observamos que se puede filtrar la búsqueda, añadir un nuevo cliente o ver detalles de un cliente concreto. Si accedemos a ver detalles del cliente nos permite editarlo, así como eliminarlo. En este diagrama se ven reflejados los requisitos específicos: 3.2, 3.3, 3.4 y 3.5.



**Figura 4.2:** Diagrama de casos de uso 2

En caso de elegir ver trabajadores en el diagrama 1 el diagrama sigue según la figura 4.3. Al igual que el diagrama 2, podemos filtrar por búsqueda, añadir un nuevo trabajador o ver detalles de un trabajador concreto. También al acceder a ver detalles del trabajador nos permite editarlo o eliminarlo. En este diagrama se ven reflejados los requisitos específicos: 3.6, 3.7, 3.8 y 3.9.



**Figura 4.3:** Diagrama de casos de uso 3

Por último, la figura 4.4 nos muestra el diagrama en el caso de elegir ver servicios en el diagrama 1. Aquí también podemos filtrar la búsqueda, añadir un servicio nuevo y ver detalles de un servicio concreto, pero estas dos últimas tienen más opciones. Al añadir un nuevo servicio podemos añadir clientes y trabajadores a este. Como ocurría en los anteriores diagramas, también podemos editar los servicios, añadiendo y eliminando tanto clientes como trabajadores. También se pueden eliminar los servicios dentro de la edición. En este diagrama se ven reflejados los requisitos específicos: 3.10, 3.11, 3.12 y 3.13.

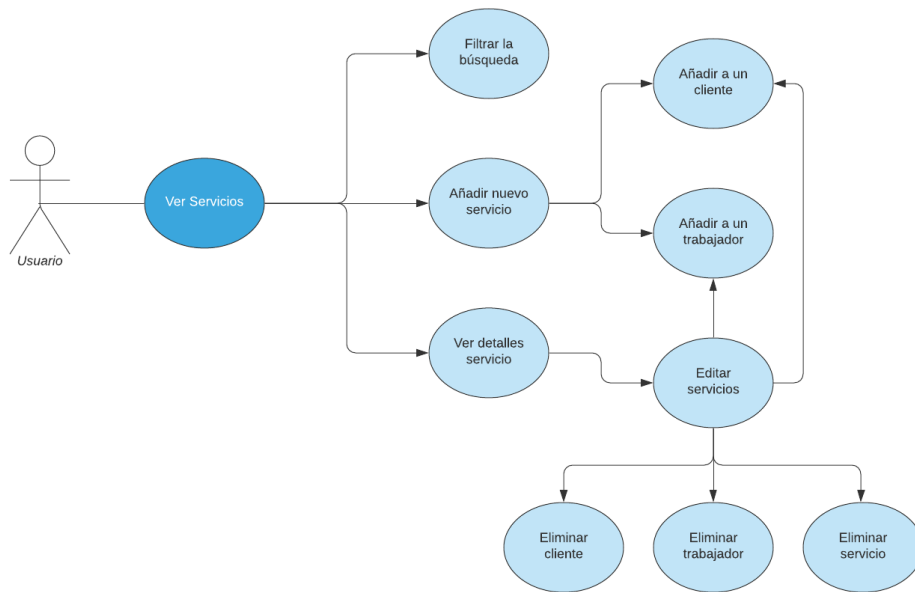


Figura 4.4: Diagrama de casos de uso 4

## 4.2 Diagrama de clases

Un diagrama de clases nos muestra como se estructura un sistema orientado a objetos. En el apartado de análisis hemos podido ver que nuestro sistema necesitará de varios objetos (clientes, trabajadores, servicios, etc.). Hay que destacar la diferencia entre un diagrama de clases y un diagrama de objetos o un diagrama de base de datos. En el caso de diagrama de objetos se utilizan ejemplos del mundo real para su representación. Un diagrama de base de datos representa atributos y relaciones a nivel de base de datos, el diagrama de clases es a nivel de la capa lógica. Odoo ya se encarga de transformar los objetos Python a PostgreSQL, así como cualquier proceso de modificación de base de datos.

En este apartado también se siguen los estándares UML.

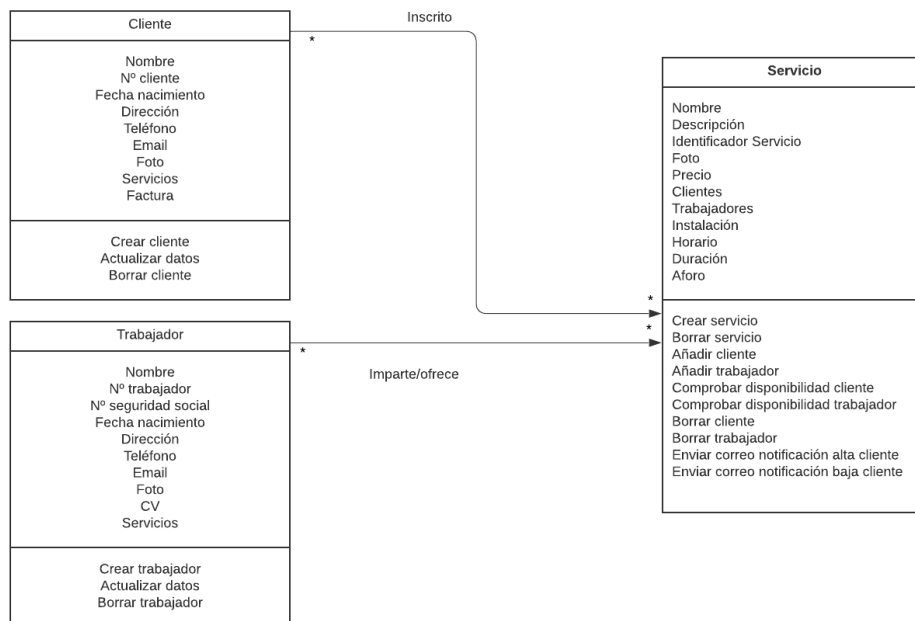


Figura 4.5: Diagrama de clases

Como vemos en la figura 4.5 hay tres clases básicas: cliente, trabajador y servicio, con sus respectivos atributos y funciones.

Cliente ofrece una serie de atributos como datos personales, email y foto. Tiene un identificador único, servicios inscritos y una factura de estos. Tiene funciones de creación de cliente, borrado de cliente y actualización de atributos.

Trabajador tiene atributos similares y también tiene identificador único. Consta de las mismas funciones anteriores pero aplicadas al trabajador.

La clase servicio es más extensa, ya que alberga más atributos y funciones. Cada servicio tiene su nombre y identificador único, algunos atributos son descriptivos, pero en cambio otros como el precio o el horario son necesarios a la hora de enlazarse con clientes o trabajadores. Al igual que las clases anteriores se puede crear, editar o borrar un servicio, pero aquí añadimos la funcionalidad de inscribir/eliminar clientes y asignar/borrar trabajadores. En estas operaciones se realizan las comprobaciones de compatibilidad pertinentes, así como notificaciones por email.

### 4.3 Prototipado

En este último apartado del diseño vemos el prototipado de la interfaz de usuario. Mediante el software Pencil se realizan una serie de bocetos o *sketches* de las diferentes ventanas a implementar en Odoo. Esto nos facilitará el futuro trabajo de desarrollo de las vistas, en el que utilizaremos las opciones en XML que Odoo nos proporciona.



Figura 4.6: Boceto vista principal

La vista principal 4.6 es sencilla y nos permite acceder a las tres partes principales de la aplicación: la gestión de clientes, trabajadores y servicios. Esta vista se puede mejorar a nivel visual con decoración que ofrece la herramienta.

#### 4.3.1. Vistas Clientes



Figura 4.7: Boceto clientes

Una vez accedemos a gestionar los clientes aparece una ventana con todos ellos [4.8](#). Existen los botones para buscar, más filtros y botón de creación de un nuevo cliente. Vemos un listado de clientes, con un *scroll* para descender y ascender. De cada cliente vemos información básica, pero podemos ver más información de cada uno.

El formulario, titulado "Nuevo Cliente", está dividido en dos secciones principales. A la izquierda, hay un espacio reservado para una fotografía, representado por un cuadrado con una 'X' diagonal y el texto "165 x 180" en el centro. Debajo de este espacio hay un botón que dice "Adjuntar foto". A la derecha, se encuentran los campos de entrada de datos, cada uno con su etiqueta correspondiente: "Nombre:" con el valor "Cliente 1", "Nº Cliente:" con "0001", "Fecha nacimiento:" con "01/01/1990", "Dirección" con "C/ Mayor nº 1", "Teléfono" con "666 666 666" y "Email" con "cliente1@gmail.com". En la parte inferior central del formulario hay un botón que dice "Crear cliente".

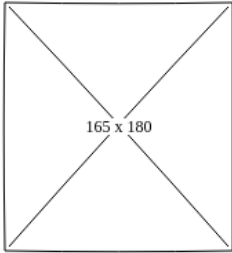
**Figura 4.8:** Boceto nuevo cliente

Al pulsar el botón Nuevo Cliente accedemos a la vista [4.8](#). Aquí rellenamos los datos necesarios y pulsamos Crear Cliente.

Si decidimos Ver más sobre un cliente en concreto la vista que aparece es la [4.9](#). Vemos que aparecen todos sus datos, editables y actualizables mediante el botón Actualizar cambios. También podemos eliminar el cliente. El sistema muestra los servicios a los que está inscrito, así como el total mensual pagado o no.



### Detalles Cliente



165 x 180

Adjuntar foto

Nombre:

Nº Cliente:

Fecha nacimiento:

Dirección:

Teléfono:

Email:

### Servicios Inscrito

ID Servicio	Nombre Servicio	Horario	Instalación	Precio
0001	Curso Natación	Lunes y Miercoles 17.00 - 18.00	Piscina	30 €/mes
0007	Suscripción Gimnasio	Lunes a Sábado 08.00 - 22.00	Gimnasio	50 €/mes

Total mensual: **70€** Pagado:  Si  No

Figura 4.9: Boceto detalles cliente

### 4.3.2. Vistas Trabajadores

Las vistas de trabajador son iguales y se accede de la misma forma que las de cliente, cambia los datos del trabajador 4.12 y que en Detalles Trabajador no nos muestra precios.

### Trabajadores

N° Trabajador	Nombre	Foto	Detalles
0001	Trabajador 1	74 x 74 	<input type="button" value="Ver más"/>
0002	Trabajador 2	74 x 74 	<input type="button" value="Ver más"/>
0003	Trabajador 3	74 x 74 	<input type="button" value="Ver más"/>
0004	Trabajador 4	74 x 74 	<input type="button" value="Ver más"/>
0005	Trabajador 5	74 x 74 	<input type="button" value="Ver más"/>
0006	Trabajador 6	74 x 74 	<input type="button" value="Ver más"/>

Figura 4.10: Boceto trabajadores

### Nuevo Trabajador

165 x 180

Nombre:

N° Trabajador:

N° SS:

Fecha nacimiento:

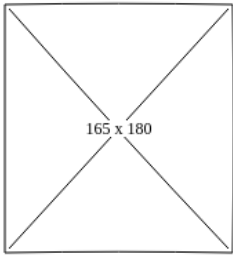
Dirección:

Teléfono:

Email:

Figura 4.11: Boceto nuevo trabajador

### Detalles Trabajador



165 x 180

Adjuntar foto

Nombre:

Nº Trabajador:

Nº SS:

Fecha nacimiento:

Dirección:

Teléfono:

Email:

Visualizar CV

Adjuntar CV

ID Servicio	Nombre Servicio	Horario	Instalación
0001	Curso Natación	Lunes y Miercoles 17.00 - 18.00	Piscina
0002	Curso Natación	Lunes y Miercoles 18.00 - 19.00	Piscina
0003	Curso Natación	Lunes y Miercoles 19.00 -20.00	Piscina
0004	Curso Natación	Martes y Jueves 17.00 - 18.00	Piscina
0005	Curso Natación	Martes y Jueves 18.00 - 19.00	Piscina
0006	Curso Natación	Martes y Jueves 19.00 - 20.00	Piscina

Actualizar cambios

Eliminar trabajador

**Figura 4.12:** Boceto detalles trabajador

### 4.3.3. Vistas Servicios

ID Servicio	Nombre	Horario	Instalación	Precio	Detalles
0001	Curso Natación	Lunes y Miércoles 17.00 - 18.00	Piscina	30 €/mes	Ver más
0002	Curso Natación	Lunes y Miércoles 18.00 - 19.00	Piscina	30 €/mes	Ver más
0003	Curso Natación	Lunes y Miércoles 19.00 - 20.00	Piscina	30 €/mes	Ver más
0004	Curso Natación	Martes y Jueves 17.00 - 18.00	Piscina	30 €/mes	Ver más
0005	Curso Natación	Martes y Jueves 18.00 - 19.00	Piscina	30 €/mes	Ver más
0006	Curso Natación	Martes y Jueves 19.00 - 20.00	Piscina	30 €/mes	Ver más
0007	Suscripción Gimnasio	Lunes a Sábado 8.00-22.00	Gimnasio	50 €/mes	Ver más

Figura 4.13: Boceto servicios

Las vistas de Servicios son más detalladas y con más opciones. Empezando por el listado general de servicios 4.13 vemos bastantes atributos, también tenemos la opción de ver más detalles. En la parte superior nos permite filtrar la búsqueda y añadir un nuevo servicio.

### Nuevo Servicio

165 x 180

Nombre:

ID Servicio:

Precio mensual:

Instalación:

Duración (minutos):

Aforo:

Descripción:

Horario:

Día

Lunes

Martes

Miércoles

Jueves

Viernes

Hora

17.00-17.30

17.30-18.00

18.00-18.30

18.30-19.00

19.00-19.30

Trabajadores:

Trabajador 1

Trabajador 2

Trabajador 3

Trabajador 4

Trabajador 5

Trabajador 6

Clientes:

Cliente 1

Cliente 2

Cliente 3

Cliente 4

Cliente 5

Cliente 6

Figura 4.14: Boceto nuevo servicio

En la creación de un nuevo servicio [4.14](#) añadimos todos los datos necesarios. La entrada del horario se realiza mediante dos desplegables, en uno seleccionamos los días y en el otro las horas. Opcionalmente podemos añadir trabajadores y clientes a este servicio.

### Detalles Servicio

165 x 180

Nombre:

ID Servicio:

Precio mensual:

Instalación:

Duración (minutos):

Aforo:

Descripción:

Horario:

Día	Horario
Lunes <input checked="" type="checkbox"/>	17.00-17.30 <input checked="" type="checkbox"/>
Martes <input type="checkbox"/>	17.30-18.00 <input checked="" type="checkbox"/>
Miércoles <input checked="" type="checkbox"/>	18.00-18.30 <input type="checkbox"/>
Jueves <input type="checkbox"/>	18.30-19.00 <input type="checkbox"/>
Viernes <input type="checkbox"/>	19.00-19.30 <input type="checkbox"/>

Trabajadores:

Trabajador 1
Trabajador 2
Trabajador 3
Trabajador 4
Trabajador 5
Trabajador 6

Clientes:

Cliente 1
Cliente 2
Cliente 3
Cliente 4
Cliente 5
Cliente 6

Figura 4.15: Boceto detalles servicio

Por último, en Detalles Servicio 4.15 vemos lo mismo que en 4.14 , solo que ahora los datos ya están introducidos. Los datos que vemos son actualizables, así como añadir o eliminar trabajadores o clientes.

---

## CAPÍTULO 5

# Desarrollo

---

En este apartado se desarrolla la aplicación Odoo aplicando lo anteriormente analizado y diseñado. También nos podemos referir a la aplicación como módulo o *addon*, y se compone de un conjunto de directorios que contienen todas las partes necesarias para su funcionamiento. En la imagen 5.1 podemos ver los directorios que componen nuestro módulo, que se explican a continuación:

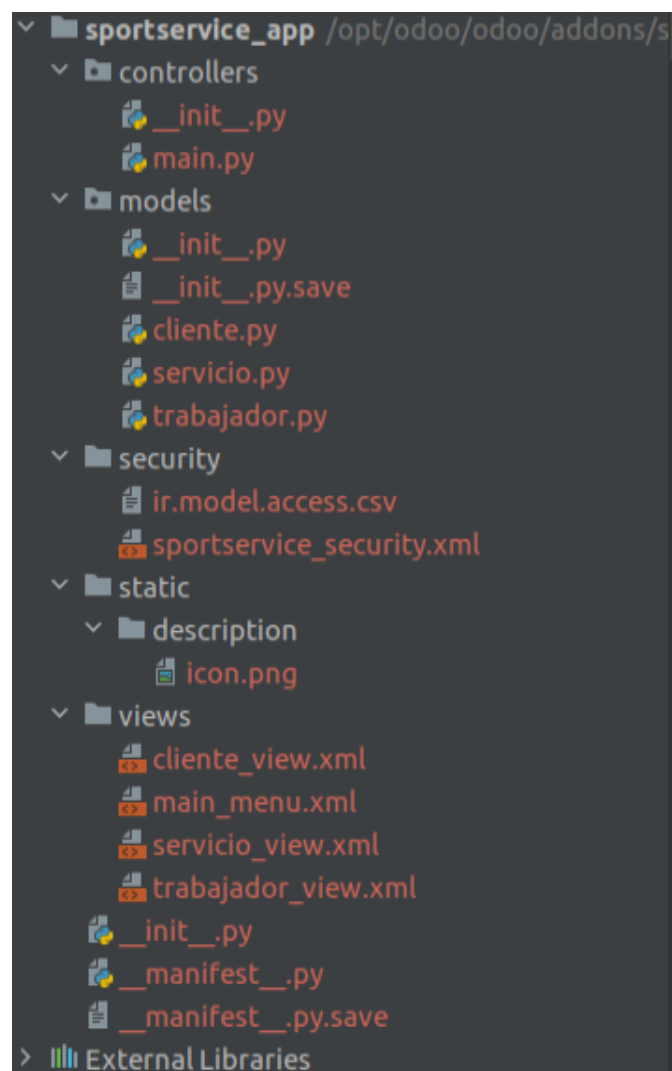


Figura 5.1: Directorios módulo

- **sportservice\_app**: Directorio que contiene todos los subdirectorios y archivos de nuestro addon. Su nombre es el que utiliza Odoon para instalar o actualizar este.
- **controllers**: Directorio que contiene archivos para desarrollar una web relacionada con el *backend* de este módulo.
- **models**: Directorio que contiene los objetos de nuestro módulo, llamados en Odoon modelos.
- **security**: Directorio que contiene la configuración de seguridad del módulo.
- **static**: Directorio que contiene datos estáticos, como por ejemplo un icono para la aplicación, archivos CSS o JavaScript, datos para importar en CSV, entre otras muchas cosas.
- **views**: Como su nombre indica, directorio con las vistas de nuestra aplicación en formato XML.

En los siguientes apartados vemos en profundidad cada uno de los archivos del módulo, explicando su función, estructura y contenido, así como mostrando algún ejemplo de como está implementado el código.

## 5.1 Modelos

A grandes rasgos los modelos son clases Python que implementan los objetos de nuestra aplicación. Odoon ya tiene implementados algunos, como por ejemplo *partner*, *client*, *report*. Se pueden utilizar estas clases, o modificarlas mediante herencia. En nuestro caso hemos creado tres modelos: cliente, trabajador y servicio.

Odoon utiliza mapeo objeto-relacional, (ORM, siglas en inglés *Object-Relational Mapping*). Es un mecanismo para enlazar clases de un lenguaje de programación orientado a objetos a un lenguaje relacional de base de datos con persistencia. En Odoon los modelos Python se traducen automáticamente a PostgreSQL, sin realizar ningún tipo de modificación a nuestra base de datos. Esto nos ayuda a la programación y a la seguridad, ya que el acceso a la base de datos es más seguro.

El sistema Odoon incluye algunos atributos por defecto, como por ejemplo un número identificador, usuario que crea el objeto, fecha de creación, fecha de modificación, etcétera. Estos atributos no se declaran.

Empezamos por el archivo `__init__.py`, que se trata de un inicializador de los módulos, por lo tanto importa los tres desarrollados:

```
1 from . import cliente , trabajador , servicio
```

El archivo `cliente.py` contiene los siguientes atributos y métodos del objeto cliente:

- **\_name** : nombre de la clase o modelo.
- **\_description** : descripción del modelo.
- **nombre** : nombre propio del cliente. Atributo obligatorio.
- **apellidos** : apellidos del cliente. Atributo obligatorio.
- **aniversario** : fecha de nacimiento del cliente.



- **direccion** : domicilio del cliente.
- **email** : dirección de correo electrónico del cliente.
- **telf** : teléfono del cliente.
- **imagen** : foto de perfil que podemos adjuntar.
- **servicios\_ids** : conjunto de servicios a los que está inscrito el cliente concreto. Se trata de un objeto *Many2many*, lo que significa que un cliente puede estar en muchos servicios y un servicio puede constar de varios clientes.
- **pagado** : atributo para anotar si un cliente ha pagado o no la cuota. Por defecto aparece como no pagado.
- **total\_servicios** : cantidad total que debe abonar el cliente por los servicios a los que está inscrito, se calcula automáticamente.
- **\_total\_servicios**: función de dependencia que calcula el atributo anterior sumando el coste de cada servicio del cliente.

Seguimos con el archivo **trabajador.py**, que contiene los siguientes atributos del objeto trabajador:

- **\_name** : nombre de la clase o modelo.
- **\_description** : descripción del modelo.
- **nombre** : nombre propio del trabajador. Atributo obligatorio.
- **apellidos** : apellidos del trabajador. Atributo obligatorio.
- **nss** : número de seguridad social del trabajador.
- **aniversario** : fecha de nacimiento del trabajador.
- **telf** : teléfono del trabajador.
- **imagen** : foto de perfil que podemos adjuntar.
- **cv** : *Curriculum Vitae* del trabajador que podemos adjuntar.
- **servicios\_ids** : conjunto de servicios a los que está inscrito el trabajador concreto. Se trata de un objeto *Many2many*, lo que significa que un trabajador puede estar en muchos servicios y un servicio puede constar de varios trabajadores.

Por último la clase más importante: **servicio.py**. Aquí encontramos los atributos que debe tener un servicio, así como funciones para el correcto funcionamiento de la aplicación:

- **\_name** : nombre de la clase o modelo.
- **\_description** : descripción del modelo.
- **nombre** : nombre del servicio. Atributo obligatorio.
- **descripcion** : descripción del servicio.
- **imagen** : icono representativo del servicio que podemos adjuntar.

- **inst** : instalaciones dónde se realiza el servicio. Atributo obligatorio.
- **precio** : precio mensual del servicio. Atributo obligatorio.
- **trabajadores\_ids** : trabajadores asignados a este servicio. Se trata de un objeto *Many2many*, lo que significa que un servicio puede tener más de un trabajador, y un trabajador puede estar en más de un servicio.
- **clientes\_ids** : clientes inscritos a este servicio. Se trata de un objeto *Many2many*, lo que significa que un servicio puede tener más de un cliente, y un cliente puede estar en más de un servicio.
- **aforo** : aforo máximo que permite el servicio. Atributo obligatorio.
- **aforoActual** : número de clientes inscritos a este servicio actualmente. Se calcula automáticamente.
- **diasSemana** : Días de la semana que se ofrece el servicio. Atributo obligatorio.
- **horario** : Horario del servicio. Atributo obligatorio.
- **aforo\_actual** : función de dependencia para calcular el número de clientes actual.
- **aforo\_valido** : función que comprueba que el aforo es el correcto, es decir, si su valor es correcto y el número de inscritos no supera el máximo.
- **horario\_cliente** : función que comprueba si hay coincidencias de horario con otros servicios de un cliente, permitiendo o no su inscripción a este servicio.
- **horario\_trabajador** : función que comprueba si hay coincidencias de horario con otros servicios de un trabajador, permitiendo o no su inscripción a este servicio.
- **correo\_alta** : función que tras una correcta inscripción a un servicio de un cliente le envía un correo electrónico automático, si el atributo email esta disponible.
- **correo\_baja** : función que tras una correcta eliminación de un cliente en un servicio le envía un correo electrónico automático, si el atributo email esta disponible.

Añadir que también hay implementadas restricciones con excepciones para los atributos que se consideren necesarios, como por ejemplo que el número de teléfono introducido es válido. Esto evita errores de introducción de datos erróneos o de seguridad. Hay que importar la librería *exceptions* para su funcionamiento. Aquí un ejemplo del teléfono:

```

1 @api.constrains('telf')
2 def checkTelf(self):
3     if len(self.telf) != 9:
4         raise ValidationError('Numero invalido!')
```

## 5.2 Vistas

En la carpeta *views* encontramos todas las vistas que componen nuestro módulo en formato XML, que el sistema Odoo se encarga de transformar a HTML para presentar los datos. Se editan independientemente del modelo que representan, son flexibles y altamente personalizables. Existen muchos tipos, *form*, *list*, *kanban*, *calendar*, *gantt*, *graph*, *Map*, *Search*, *QWeb*, nosotros definimos una vista *form* (formulario) y *list* (listado) para cada modelo. A continuación se explica cada uno de los archivos que componen el directorio.

- **main\_menu.xml** : contiene el menú principal de la aplicación. Por cada modelo definimos una acción y un menú que aparece al seleccionar dicha acción. Las distintas acciones tienen un nombre, un enlace al modelo y el tipo de vista. Los distintos menú items tienen un nombre, un menú padre (en este caso los tres tienen como padre al principal) y una acción correspondiente.

```

1  <?xml version="1.0"?>
2  <odoo>
3    <!-- SportService App Menu -->
4    <menuitem id="main_menu"
5              name="SportService" />
6    <!-- Abrir listado de clientes -->
7    <act_window id="action_listado_clientes"
8               name="Listado Clientes"
9               res_model="sportservice.cliente"
10              view_mode="tree,form"
11             />
12    <!-- Menu item para abrir el listado de clientes -->
13    <menuitem id="listado_clientes"
14              name="Clientes"
15              parent="main_menu"
16              action="action_listado_clientes"
17             />
18    <!-- Abrir listado de trabajadores -->
19    <act_window id="action_listado_trabajadores"
20               name="Listado Trabajadores"
21               res_model="sportservice.trabajador"
22              view_mode="tree,form"
23             />
24    <!-- Menu item para abrir el listado de trabajadores -->
25    <menuitem id="listado_trabajadores"
26              name="Trabajadores"
27              parent="main_menu"
28              action="action_listado_trabajadores"
29             />
30    <!-- Abrir listado de clientes -->
31    <act_window id="action_listado_servicios"
32               name="Listado Servicios"
33               res_model="sportservice.servicio"
34              view_mode="tree,form"
35             />
36    <!-- Menu item para abrir el listado de clientes -->
37    <menuitem id="listado_servicios"
38              name="Servicios"
39              parent="main_menu"
40              action="action_listado_servicios"
41             />
42  </odoo>

```

- **cliente\_view.xml** : contiene las vistas para el módulo cliente. Vemos dos registros (*record*) diferentes con sus respectivos identificadores. El primero nos muestra el menú para cada cliente, tanto para visualizar como editar clientes. Vemos que están enumerados la mayoría de sus atributos. El segundo se trata de la vista de árbol, es decir, como aparecen nuestro modelo en una lista. Nuestra aplicación utiliza listas de sus modelos en muchas ocasiones, y aquí definimos como se ve.

```

1  <?xml version="1.0"?>
2  <odoo>
3    <record id="view_form_cliente" model="ir.ui.view">
4      <field name="name">Cliente</field>
5      <field name="model">sportservice.cliente</field>
6      <field name="arch" type="xml">

```

```

7         <form string="Cliente">
8             <group>
9                 <field name="imagen" widget="image" height="50"/>
10                <field name="id" />
11                <field name="nombre" />
12                <field name="apellidos" />
13                <field name="aniversario" />
14                <field name="direccion" />
15                <field name="email" />
16                <field name="telf" />
17                <field name="servicios_ids"/>
18                <field name="total_servicios" />
19                <field name="pagado" />
20            </group>
21        </form>
22    </field>
23 </record>
24 <record id="view_tree_cliente" model="ir.ui.view">
25     <field name="name">Lista Clientes</field>
26     <field name="model">sportservice.cliente</field>
27     <field name="arch" type="xml">
28         <tree>
29             <field name="id" />
30             <field name="nombre" />
31             <field name="apellidos" />
32             <field name="imagen" widget="image" height="50"/>
33         </tree>
34     </field>
35 </record>
36 </odoo>

```

- **trabajador\_view.xml** : contiene las vistas para el módulo trabajador. Las vistas son similares a las de cliente, cambiando lógicamente los atributos para el modelo trabajador. Se define también tanto su menú como su vista de árbol.

```

1 <?xml version="1.0"?>
2 <odoo>
3     <record id="view_form_trabajador" model="ir.ui.view">
4         <field name="name">Trabajador</field>
5         <field name="model">sportservice.trabajador</field>
6         <field name="arch" type="xml">
7             <form string="Trabajador">
8                 <group>
9                     <field name="nombre" />
10                    <field name="apellidos" />
11                    <field name="nss" />
12                    <field name="aniversario" />
13                    <field name="telf" />
14                    <field name="imagen" widget="image" height="50"/>
15                    <field name="cv" />
16                    <field name="servicios_ids" />
17                </group>
18            </form>
19        </field>
20    </record>
21    <record id="view_tree_trabajador" model="ir.ui.view">
22     <field name="name">Lista Trabajadores</field>
23     <field name="model">sportservice.trabajador</field>
24     <field name="arch" type="xml">
25         <tree>
26             <field name="id" />
27             <field name="nombre" />
28             <field name="apellidos" />
29             <field name="imagen" widget="image" height="50"/>

```

```

30     </tree>
31     </field>
32 </record>
33 </odoo>

```

- **servicio\_view.xml** : contiene las vistas para el módulo servicio. Se estructura como las vistas anteriores, pero añadiendo más atributos, ya que el módulo servicio es más complejo. Destacamos el uso de grupos para distribuir mejor el contenido que se muestra, y una barra de progreso del aforo.

```

1  <?xml version="1.0"?>
2  <odoo>
3    <record id="view_form_servicio" model="ir.ui.view">
4      <field name="name">Servicio</field>
5      <field name="model">sportservice.servicio</field>
6      <field name="arch" type="xml">
7        <form string="Servicio">
8          <group>
9            <field name="imagen" widget="image" height="50"/>
10           <field name="nombre" />
11           <field name="descripcion" />
12           <field name="inst" />
13           <field name="precio" />
14           <field name="diasSemana" />
15           <field name="horario" />
16           <field name="aforo" />
17           <field name="aforoActual" widget="progressbar"/>
18         </group>
19         <group>
20           <group>
21             <field name="trabajadores_ids" />
22           </group>
23           <group>
24             <field name="clientes_ids" />
25           </group>
26         </group>
27       </form>
28     </field>
29 </record>
30 <record id="view_tree_servicio" model="ir.ui.view">
31   <field name="name">Lista Servicios</field>
32   <field name="model">sportservice.servicio</field>
33   <field name="arch" type="xml">
34     <tree>
35       <field name="id" />
36       <field name="nombre" />
37       <field name="imagen" widget="image" height="50"/>
38       <field name="inst" />
39       <field name="diasSemana" />
40       <field name="horario" />
41       <field name="precio" />
42     </tree>
43   </field>
44 </record>
45 </odoo>

```

## 5.3 Seguridad

El directorio de seguridad es también un elemento importante en cualquier addon. Aquí se configuran los grupos para configurar los controles de acceso y los permisos me-

diante una lista ACL (en inglés *Access-Control List*). Cada modelo debe tener sus propias reglas de acceso a los respectivos grupos de usuarios.

El archivo **sportservice\_security.xml** es para la creación de los grupos de usuarios de nuestro addon. A continuación podemos ver como se definen los dos grupos de usuarios (*records*):

- **sportservice\_group\_user** : grupo de permisos básicos del grupo group\_user de Odoo.
- **sportservice\_group\_manager**: grupo de permisos de los grupos group\_user, user\_root y user\_admin de Odoo. Grupo más privilegiado que el anterior por defecto.

```

1 <?xml version="1.0" ?>
2 <odoo>
3   <record id="module_sportservice_category" model="ir.module.category">
4     <field name="name">SportService</field>
5   </record>
6   <!-- SportService User Group -->
7   <record id="sportservice_group_user" model="res.groups">
8     <field name="name">User</field>
9     <field name="category_id"
10        ref="module_sportservice_category"/>
11     <field name="implied_ids"
12        eval="[(4, ref('base.group_user'))]"/>
13   </record>
14   <!-- SportService Manager Group -->
15   <record id="sportservice_group_manager" model="res.groups">
16     <field name="name">Manager</field>
17     <field name="category_id"
18        ref="module_sportservice_category"/>
19     <field name="implied_ids"
20        eval="[(4, ref('sportservice_group_user'))]"/>
21     <field name="users"
22        eval="[(4, ref('base.user_root')),
23              (4, ref('base.user_admin'))]"/>
24   </record>
25 </odoo>

```

El archivo **ir.model.access.csv** almacena los permisos de cada grupo definido anteriormente en cada modelo. Los campos del archivo CSV son los siguientes:

- **id** : identificador único del registro XML.
- **name** : nombre descriptivo.
- **model\_id** : identificador del modelo externo al que estamos configurando el acceso.
- **group\_id** : identificador del grupo de seguridad al que se le esta configurando el acceso.
- **perm\_read** : permiso de lectura, 0 desactivado 1 activado.
- **perm\_write** : permiso de escritura, 0 desactivado 1 activado.
- **perm\_create** : permiso de creación, 0 desactivado 1 activado.
- **perm\_unlink** : permiso de borrado, 0 desactivado 1 activado.

```

1 id ,name, model_id : id , group_id : id , perm_read , perm_write , perm_create , perm_unlink
2 access_cliente_user , clienteUser , model_sportservice_cliente ,
   sportservice_group_user , 1 , 0 , 0 , 0
3 access_cliente_manager , clienteManager , model_sportservice_cliente ,
   sportservice_group_manager , 1 , 1 , 1 , 1
4 access_trabajador_user , trabajadorUser , model_sportservice_trabajador ,
   sportservice_group_user , 1 , 0 , 0 , 0
5 access_trabajador_manager , trabajadorManager , model_sportservice_trabajador ,
   sportservice_group_manager , 1 , 1 , 1 , 1
6 access_servicio_user , servicioUser , model_sportservice_servicio ,
   sportservice_group_user , 1 , 0 , 0 , 0
7 access_servicio_manager , servicioManager , model_sportservice_servicio ,
   sportservice_group_manager , 1 , 1 , 1 , 1

```

En la tabla 5.1 podemos ver como quedan los permisos en los grupos de acceso a los modelos de nuestra aplicación.

Modelo	Lectura		Escritura		Creación		Borrado	
	Manager	Usuario	Manager	Usuario	Manager	Usuario	Manager	Usuario
Cliente	Si	Si	Si	No	Si	No	Si	No
Trabajador	Si	Si	Si	No	Si	No	Si	No
Servicio	Si	Si	Si	No	Si	No	Si	No

**Tabla 5.1:** Tabla resumen de permisos

Durante el desarrollo y pruebas del proyecto se ha utilizado el usuario manager del ERP completo. Para un uso real se puede crear un nuevo usuario y asignar a dicho usuario un grupo. El grupo determinará el nivel de seguridad, en caso de pertenecer a varios grupos el sistema elige al que tiene más privilegios. Para acceder a los usuarios debemos de acceder a Odoos como desarrollador, e ir a Dashboard/Users. Aquí podemos crear nuevos usuarios, o editar los actuales. En la imagen 5.2 podemos ver al usuario administrador, que es Manager para la aplicación SportService. Podemos editar el usuario para, por ejemplo, degradar de User con menos permisos.

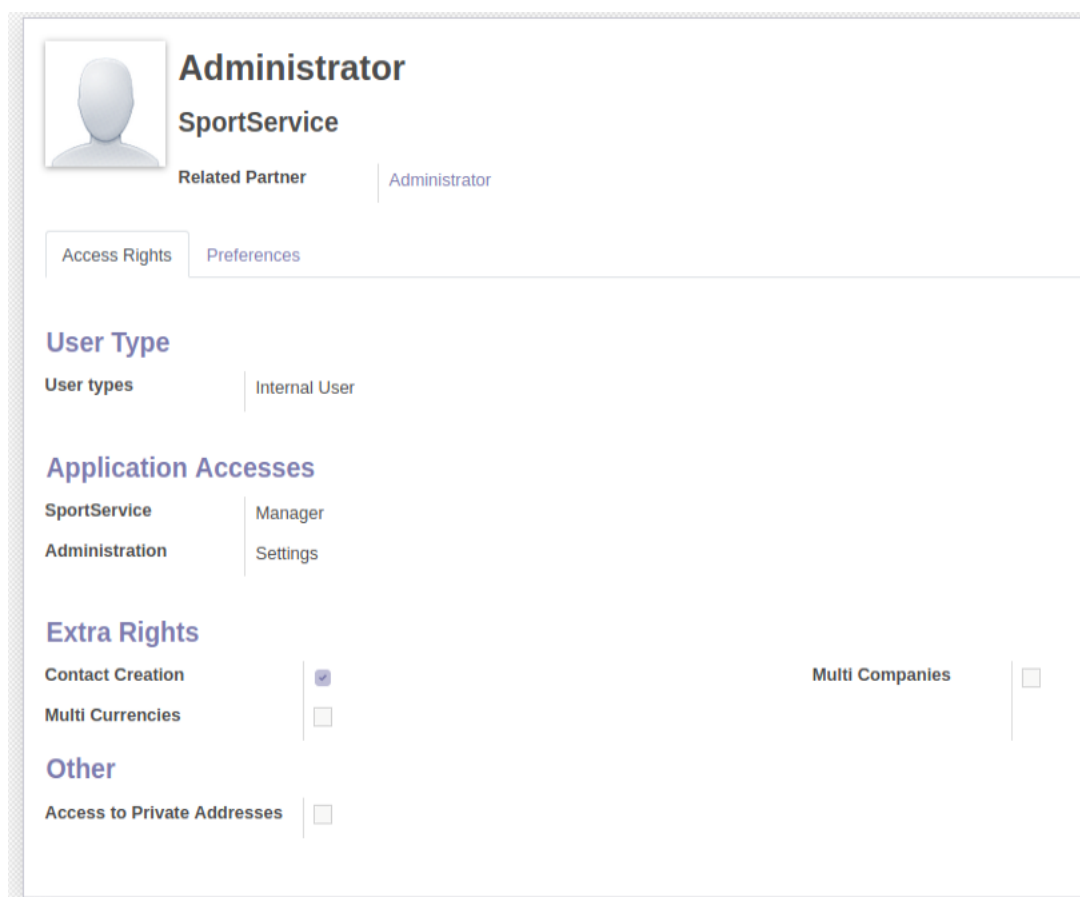


Figura 5.2: Usuario Administrador

## 5.4 Otros archivos

Por último, en este apartado se comentan otros archivos que no se agrupan en los directorios anteriores.

- **\_\_init\_\_.py** : como ya hemos visto en otros apartados, se trata de un archivo descriptor de módulos Python. Funciona como un módulo de Python que se ejecuta al arrancar el programa. Ayuda a la gestión de paquetes y archivos que hay que cargar a Odoo, en este caso necesarios para la ejecución de la aplicación SportService. Importamos los modelos creados en el primer apartado.

```
1 from . import models
```

- **\_\_manifest\_\_.py** : sirve para especificar los metadatos del módulo y el aspecto que tiene el módulo dentro de la lista de aplicaciones de Odoo. Podemos observar el resultado en la imagen 5.3. Todos los metadatos están representados mediante un diccionario Python. En el campo data debemos incluir todas las vistas y archivos de seguridad, para que se carguen al iniciar la aplicación. Este es el contenido del archivo en cuestión:

```
1 {
2     'name': 'SportService',
3     'description': 'Aplicacion de gestion de servicios deportivos.',
4     'author': 'Pau Sastre Pons',
5     'depends': ['base'],
```



```
6     'application': True ,
7     'data': [
8         'security/sportservice_security.xml' ,
9         'security/ir.model.access.csv' ,
10        'views/main_menu.xml' ,
11        'views/cliente_view.xml' ,
12        'views/trabajador_view.xml' ,
13        'views/servicio_view.xml' ,
14    ],
15 }
```

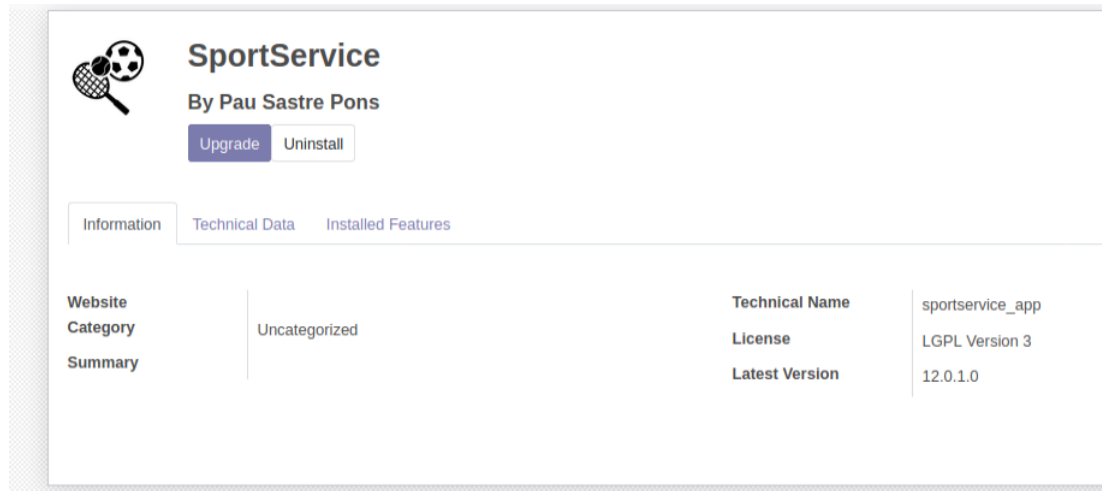


Figura 5.3: Aplicación SportService

- **controllers** : archivo para desarrollar un portal web con los datos *backend* del módulo. No se ha utilizado para este proyecto, pero se ha creado para realizar pruebas. En un futuro se puede desarrollar un aplicación web fácilmente con el framework que Odoo nos ofrece.
- **static** : aquí se almacenan datos estáticos para la aplicación. En nuestro caso almacenamos el icono de la aplicación, pero tiene muchos más usos. Podemos añadir archivos CSV para importar datos, por ejemplo rellenar de golpe ciertos módulos. Otro uso es para archivos Javascript o CSS, en caso de utilizar interfaces más complejas en las vistas.



# Implantación y mantenimiento

---

En este apartado se muestra como configurar y arrancar Odoo en un sistema basado en Debian, lo más parecido posible a un entorno de producción. Se trabaja sobre los archivos de configuración, para utilizar varios procesos, securizar o escalar el servicio, entre otras cosas. También se explica brevemente como configura un proxy inverso *nginx* y la configuración del HTTPS, tanto para mejorar como proteger el servidor. Para finalizar veremos como realizar copias de seguridad y actualizaciones del servidor y sus módulos.

## 6.1 Configuración general

---

Para empezar crearemos el archivo de configuración que utilizará el servidor Odoo. Las siguientes ordenes de terminal crean y securizan de forma básica el archivo, y lo situamos en `/etc/odoo`:

```
1 $ sudo su -c "~/odoo/odoo-bin -d odoo-prod" \ --db-filter='^odoo-prod$' --
   without-demo=all" \ -i base --save --stop-after-init" odoo
2 $ sudo mkdir /etc/odoo
3 $ sudo cp /home/odoo-dev/.odoorc /etc/odoo/odoo.conf
4 $ sudo chown -R odoo /etc/odoo
5 $ sudo chmod u=r,g=rw,o=r /etc/odoo/odoo.conf
```

Mantener logs de nuestro sistema es una buena práctica, debemos configurarlo para mantener un registro de los eventos que han sucedido en nuestra máquina. Con las siguientes ordenes se almacenarán en `/var/log/odoo`:

```
1 $ sudo mkdir /var/log/odoo
2 $ sudo chown odoo /var/log/odoo
```

A continuación editaremos el archivo de configuración recién creado:

```
1 $ sudo nano /etc/odoo/odoo.conf
```

Dentro del editor de texto de la terminal nos interesan los siguientes parámetros, explicados tras la tabla:

```
1 [options]
2 addons_path = /home/odoo/odoo-12/odoo/addons,/home/odoo/odoo-12/addons
3 admin_passwd = False
4 db_name = odoo-prod
5 dbfilter = ^odoo-prod$
6 http_port = 8069
7 list_db = False
8 logfile = /var/log/odoo/odoo-server.log
9 proxy_mode = True
```

```
10 without_demo = all
11 workers = 6
```

- **addons path:** Las rutas donde se buscarán los *addons* instalados. Su orden de lectura es de izquierda a derecha, por lo tanto los de la izquierda son de mayor prioridad.
- **admin passwd:** La contraseña para administrar la base de datos del cliente web, se recomienda poner una contraseña segura o desactivar esta opción.
- **db name:** la instancia de la base de datos a inicializar durante la secuencia de inicio.
- **dbfilter:** Filtro para hacer más accesible la base de datos. Se trata de una expresión *regex* interpretada por Python.
- **http port:** El número de puerto donde escuchará el servidor, por defecto es el 8069.
- **list db:** Bloquea que se enumeren las bases de datos.
- **logfile:** Le indicamos al servidor donde debe almacenar los logs, en el directorio creado previamente.
- **proxy mode:** Activamos el modo proxy, se configura en el apartado 6.3.
- **without demo:** Recomendado para entornos de producción, para que las nuevas bases de datos no tengan datos de ejemplo dentro ocupando espacio o mostrando resultados no esperados en consultas.
- **workers:** Con un número mayor que dos activamos el modo multiproceso.

Podemos ver a tiempo real que esta ocurriendo en nuestro servidor con el siguiente comando:

```
1 $ sudo tail -f /var/log/odoo/odoo-server.log
```

## 6.2 Configuración del servicio

Una vez desarrollado y configurado el servicio debemos preparar la máquina para que arranque Odoo al iniciar el sistema operativo, ya que entendemos que será el propósito de esta. En el caso de Ubuntu, en versiones posteriores a la 16.04, el sistema *init* es el responsable de las funciones de inicio del sistema operativo.

El primer paso es crear y rellenar el archivo de inicio con las siguientes ordenes:

```
1 $ cd /lib/systemd/system/
2 $ sudo nano odoo.service
```

Rellenar y guardar los siguientes datos:

```
1 [Unit]
2 Description=Odoo
3 After=postgresql.service
4
5 [Service]
6 Type=simple
7 User=odoo
8 Group=odoo
9 ExecStart=/home/odoo/odoo-12/odoo-bin -c /etc/odoo/odoo.conf
```

```

10 |
11 | [ Install ]
12 | WantedBy=multi-user.target

```

A continuación la orden para registrar el servicio, y otras ordenes útiles:

```

1 | $ sudo systemctl enable odoo.service #registrar servicio
2 | $ sudo systemctl start odoo         #arrancar el servicio
3 | $ sudo systemctl status odoo        #comprobar si se esta ejecutando
4 | $ sudo systemctl stop odoo         #parar el servicio

```

## 6.3 Configuración del servidor inverso

Para cualquier página web que dé servicio al público es recomendable utilizar un proxy inverso delante del servidor. En Odoo no es distinto, se puede instalar uno y en nuestro caso será nginx (<https://www.nginx.com/>). Los motivos son varios, pero los principales son: la seguridad y el rendimiento.

Al estar de cara al internet público hay que protegerse, y poner un proxy en medio es una de las opciones recomendadas. Se encarga de que ls protocolos HTTPS vayan encriptados y actúa como firewall de aplicación, limitando las URLs que se aceptan. También esconde las características internas de la red.

Hablando de rendimiento, el proxy se encarga de cachear de forma estática contenido redundante, quitando así carga al servidor Odoo. Así mismo comprime el contenido para acelerar los tiempos de carga. Podría actuar de balanceador de carga en caso de tener más servidores, no es el caso.

El primer paso para instalar nginx es desactivar cualquier servicio que tenemos escuchando el puerto por defecto de HTTP, como podría ser un servidor Apache. Se debería detener o cambiar de puerto este servicio. En nuestro caso no hay nada más corriendo en la máquina, pero se tiene que tener en cuenta.

Una vez asegurado esto, con la siguiente orden instalamos y arrancamos nginx:

```

1 | $ sudo apt-get install nginx
2 | $ sudo service nginx start

```

Debemos desactivar la configuración por defecto y crear nuestra propia configuración para Odoo con las siguientes ordenes:

```

1 | $ sudo rm /etc/nginx/sites-enabled/default
2 | $ sudo touch /etc/nginx/sites-available/odoo
3 | $ sudo ln -s /etc/nginx/sites-available/odoo \
4 | /etc/nginx/sites-enabled/odoo
5 | $ sudo nano /etc/nginx/sites-available/odoo

```

En el editor de texto de la terminal introducir la siguiente configuración:

```

1 | upstream odoo {
2 |     server 127.0.0.1:8069;
3 | }
4 | upstream odoochat {
5 |     server 127.0.0.1:8072;
6 | }
7 | server {
8 |     listen 80;
9 |     proxy_set_header X-Forwarded-Host $host;
10 |    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
11 |    proxy_set_header X-Forwarded-Proto $scheme;

```

```

12 proxy_set_header X-Real-IP $remote_addr;
13 access_log /var/log/nginx/odoo.access.log;
14 error_log /var/log/nginx/odoo.error.log;
15 location /longpolling {
16     proxy_pass http://odoochat;
17 }
18 location ~* /web/static/ {
19     proxy_cache_valid 200 60m;
20     proxy_buffering on;
21     expires 864000;
22     proxy_pass http://odoo;
23 }
24 gzip_types text/css text/scss text/plain text/xml application/xml application/
    json application/javascript;
25 gzip on;
26 }

```

Ya se ha configurado la redirección de tráfico y la compresión del tráfico que puede ser comprimido. También nos cachea datos estáticos durante una hora. Por último debemos recargar la nueva configuración:

```
1 $ sudo /etc/init.d/nginx reload
```

## 6.4 Configuración del HTTPS

Hoy en día es crítico para cualquier aplicación comercial el uso de HTTPS, cuyas siglas en inglés significan *Hypertext Transfer Protocol Secure*. Básicamente se cifran los textos mediante protocolos basados en SSL/TLS. En nuestro caso para simplificar las cosas se utilizará un certificado auto-firmado, pero para otras soluciones es recomendable usar autoridades certificadoras. Odoo tiene módulos para peticiones SSL, utilizando <https://letsencrypt.org/>, pero en este caso no se utilizarán.

El primer paso consiste en crear y proteger el certificado auto-firmado:

```

1 $ sudo mkdir /etc/ssl/nginx && cd /etc/ssl/nginx
2 $ sudo openssl req -x509 -newkey rsa:2048 \
3 -keyout server.key -out server.crt -days 365 -nodes
4 $ sudo chmod a-wx *
5 $ sudo chown www-data:root *

```

Ahora debemos editar el texto creado en el apartado anterior 6.3:

```
1 $ sudo nano /etc/nginx/sites-available/odoo
```

Cambiamos la línea 8, y añadimos las líneas 13-19:

```

1 upstream odoo {
2     server 127.0.0.1:8069;
3 }
4 upstream odoochat {
5     server 127.0.0.1:8072;
6 }
7 server {
8     listen 443;
9     proxy_set_header X-Forwarded-Host $host;
10    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
11    proxy_set_header X-Forwarded-Proto $scheme;
12    proxy_set_header X-Real-IP $remote_addr;
13    ssl on;
14    ssl_certificate /etc/ssl/nginx/server.crt;

```

```

15 ssl_certificate_key /etc/ssl/nginx/server.key;
16 ssl_session_timeout 30m;
17 ssl_protocols TLSv1 TLSv1.1 TLSv1.2;
18 ssl_ciphers 'ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-
    RSA-AES256-GCM-SHA384:ECDHE-ECDSA-AES256-GCM-SHA384:DHE-RSA-AES128-GCM-
    SHA256:DHE-DSS-AES128-GCM-SHA256:kEDH+AESGCM:ECDHE-RSA-AES128-SHA256:
    ECDHE-ECDSA-AES128-SHA256:ECDHE-RSA-AES128-SHA:ECDHE-ECDSA-AES128-SHA:
    ECDHE-RSA-AES256-SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-RSA-AES256-SHA:
    ECDHE-ECDSA-AES256-SHA:DHE-RSA-AES128-SHA256:DHE-RSA-AES128-SHA:DHE-DSS-
    AES128-SHA256:DHE-RSA-AES256-SHA256:DHE-DSS-AES256-SHA:DHE-RSA-AES256-SHA
    :AES128-GCM-SHA256:AES256-GCM-SHA384:AES128-SHA256:AES256-SHA256:AES128-
    SHA:AES256-SHA:AES:CAMELLIA:DES-CBC3-SHA:!aNULL:!eNULL:!EXPORT:!DES:!RC4
    :!MD5:!PSK:!aECDH:!EDH-DSS-DES-CBC3-SHA:!EDH-RSA-DES-CBC3-SHA:!KRB5-DES-
    CBC3-SHA';
19 ssl_prefer_server_ciphers on;
20 access_log /var/log/nginx/odoo.access.log;
21 error_log /var/log/nginx/odoo.error.log;
22 location /longpolling {
23     proxy_pass http://odoochat;
24 }
25 location ~* /web/static/ {
26     proxy_cache_valid 200 60m;
27     proxy_buffering on;
28     expires 864000;
29     proxy_pass http://odoo;
30 }
31 gzip_types text/css text/scss text/plain text/xml application/xml application/
    json application/javascript;
32 gzip on;

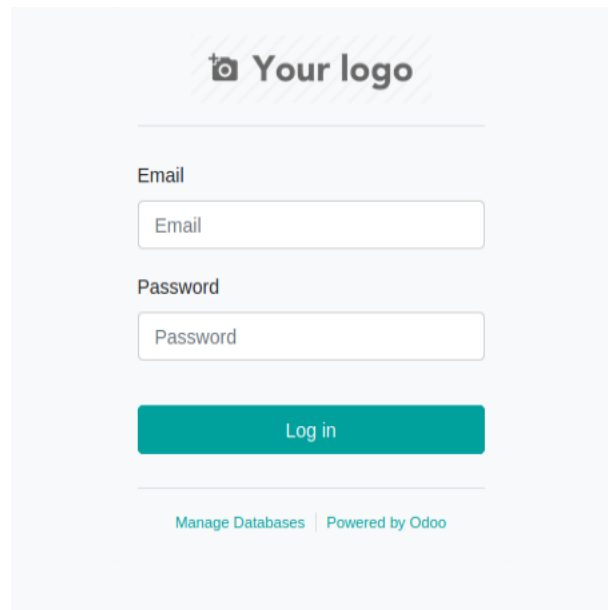
```

A partir de ahora escucha el puerto seguro HTTPS y usará el certificado creado para nginx para encriptar el tráfico.

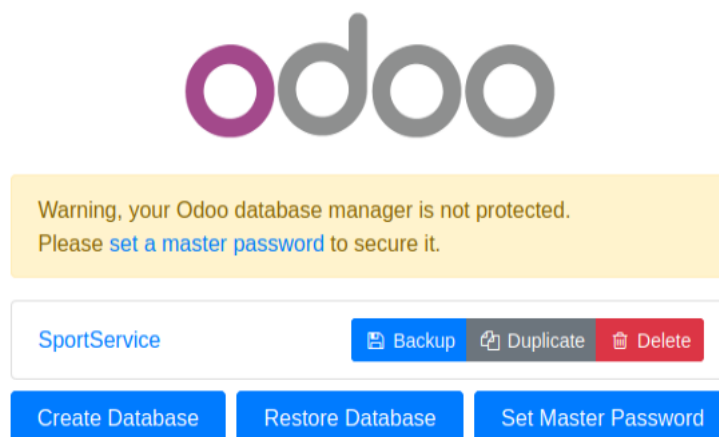
## 6.5 Copia de seguridad y gestión de bases de datos

La realización de copias de seguridad en Odoo es muy simple e intuitiva. En el menú de inicio de sesión [6.1](#) debemos acceder abajo a la izquierda: Manage Databases. Aquí accedemos al gestor de base de datos, sin uso de comandos PostgreSQL. Como vemos en la [6.2](#), nos permite realizar varias acciones:

- **Set Master Password:** Nos permite proteger todas las bases de datos mediante un usuario maestro.
- **Create Database:** Crear una base de datos nueva.
- **Restore Database:** Restaura una base de datos.
- **Backup:** Realiza una copia de seguridad de una base de datos concreta.
- **Duplicate:** Duplica una base de datos concreta.
- **Delete:** Elimina una base de datos concreta.



**Figura 6.1:** Boceto detalles servicio



**Figura 6.2:** Boceto detalles servicio

Para arrancar una base de datos concreta debemos arrancar Odoo de la siguiente manera:

```
$ ./odoo-bin -d SportService --save
```

Debemos estar situados en el directorio donde se sitúa `odoo-bin`, o escribir su ruta absoluta. Con el parámetro `-d` indicamos qué base de datos arranca nuestra instancia de Odoo. El parámetro `--save` guarda en archivos de configuración la base de datos indicada por defecto. Ambos parámetros son opcionales.

## 6.6 Instalación y actualización de módulos

En este apartado explicamos como instalar y actualizar módulos personalizados, a los que llamamos *custom addons*. Antes de empezar el diseño de la aplicación Odoo debemos



configurar el sistema para que cargue las aplicaciones personalizadas, como es nuestro caso. Para ello debemos acceder a la carpeta donde tenemos instalado Odoo y ejecutar la siguiente línea de comandos para añadir la ruta de aplicaciones personalizadas:

```
1 $ ./odoo/odoo-bin -d todo --addons-path="custom-addons,odoo/addons" --save
```

En la carpeta `custom-addons` se guarda nuestra aplicación, y con la opción `save` Odoo se acuerda de cargar módulos de esa carpeta al arrancar la próxima vez. El archivo se puede situar donde sea, siempre que tenga los permisos necesarios.

Cuando el `addon` es mínimamente funcional debemos instalarlo en nuestro sistema Odoo con el siguiente parámetro `-i`:

```
1 $ ./odoo-bin -i sportservice_app
```

Cuando se realizan cambios en un módulo debemos ejecutar el parámetro `-u` para aplicar los cambios, así como detectar posibles errores estáticos en el código:

```
1 $ ./odoo/odoo-bin -i sportservice_app
```

## 6.7 Actualización de servidor

En caso de que esté disponible una nueva versión de Odoo tenemos la opción de actualizar a esta última. Al instalar Odoo de su repositorio oficial de Github, su actualización no es una tarea muy compleja. En primer lugar debemos realizar una copia de seguridad de nuestros datos, así como de los `addons` personalizados, si es el caso. En principio esto no afecta a la actualización, pero por si ocurre algún problema durante la actualización y queremos volver atrás debemos tener *backups* por seguridad. Una vez realizada la copia accedemos al repositorio donde tenemos instalado Odoo, donde lanzamos el orden de instalación con `git`. Situados aquí lanzamos las siguientes órdenes `git`:

```
1 $ git fetch
2 $ git rebase --autostash
```

El primer comando actualiza las ramas de este directorio. El segundo nos busca conflictos en caso de haber editado el código local de Odoo. Hay que tener en cuenta que el código de Odoo se actualiza respecto al anterior, si se ha modificado manualmente este puede haber dificultades al actualizar al nuevo. Manualmente elegiremos que cambios en el código conservamos. En caso de querer ignorar estos cambios y restaurar la versión oficial lanzamos el siguiente comando:

```
1 $ git reset --hard
```

Una vez actualizado Odoo debemos detener su ejecución con `Control+C`, en caso de que el sistema estuviera en marcha, y volvemos a ejecutar como siempre (opcionalmente con los parámetros de arranque que necesitamos):

```
1 $ ./odoo-bin
```

Para más información acceder a la documentación oficial: <https://www.odoo.com/documentation/12.0/setup/update.html>.



---

## CAPÍTULO 7

# Pruebas funcionales

---

Una vez analizada, diseñada e implementada la aplicación debemos verificar el correcto funcionamiento de esta mediante casos de prueba. Un caso de prueba es una breve declaración de algo que debería ser probado. Se trata de un mecanismo que puede ser manual o automático, y verifica si el comportamiento de un sistema es el deseado o no [5]. Para verificar dicho comportamiento comprobamos que la aplicación cumple con los requisitos funcionales planteados en el apartado 3.1.3.

La primera prueba a realizar es el inicio de sesión correcto, como vemos en la figura 7.1. Dentro de Odoo podemos cerrar la sesión en cualquier momento en la esquina superior derecha 7.2. Verificamos que se cumplen los requisitos específicos 3.1 y 3.16.

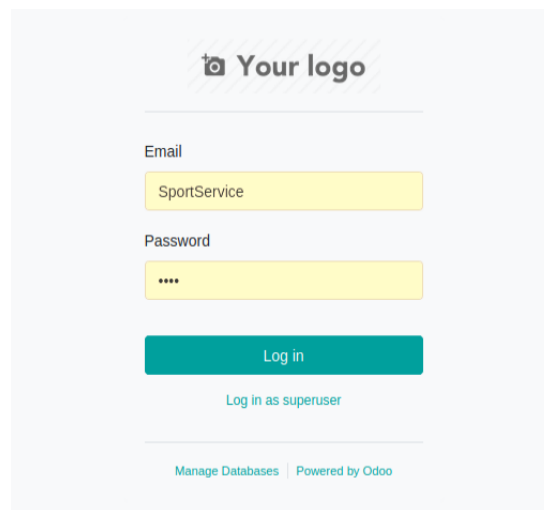


Figura 7.1: Inicio de sesión

Una vez instalada correctamente la aplicación aparece como acceso directo en el menú superior izquierdo de Odoo. Al acceder a la aplicación podemos ir a las pestañas de clientes, trabajadores o servicios, tal como podemos ver en la figura 7.3.



Figura 7.3: Menú principal

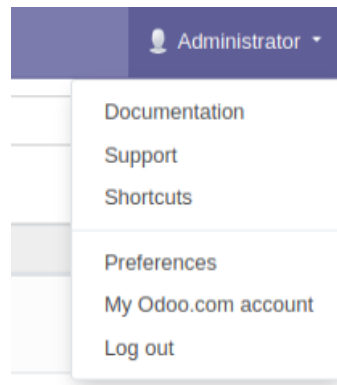
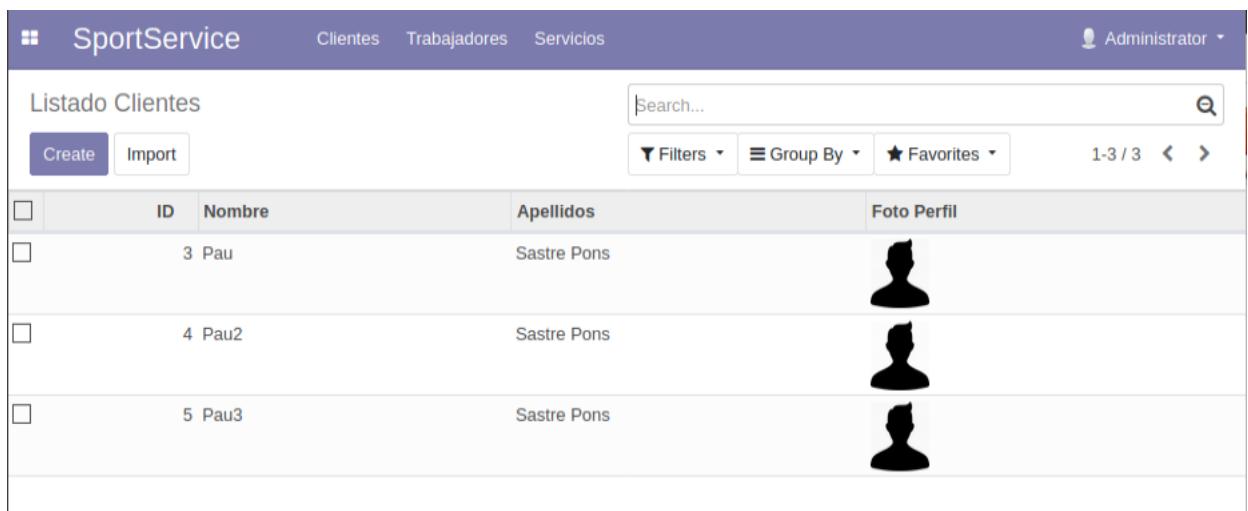


Figura 7.2: Cierre de sesión

Situados en el menú principal 7.3, al hacer clic sobre Clientes accedemos a la vista de la imagen 7.4. Verificamos que se cumple el requisito específico 3.2.



The screenshot shows the 'SportService' application interface. The top navigation bar includes 'SportService', 'Clientes', 'Trabajadores', 'Servicios', and 'Administrator'. The main content area is titled 'Listado Clientes' and features a search bar, 'Create' and 'Import' buttons, and filters for 'Filters', 'Group By', and 'Favorites'. The data is presented in a table with the following columns: ID, Nombre, Apellidos, and Foto Perfil.




ID	Nombre	Apellidos	Foto Perfil
3	Pau	Sastre Pons	
4	Pau2	Sastre Pons	
5	Pau3	Sastre Pons	

Figura 7.4: Vista Clientes

Situados nuevamente en el menú principal 7.3, al hacer clic sobre Trabajadores accedemos a la vista de la imagen 7.5. Verificamos que se cumple el requisito específico 3.6.

ID	Nombre	Apellidos	Foto Perfil
1	Pepito	Ejemplo	
2	Juanito	Ejemplo	
3	Antonio	Ejemplo	

Figura 7.5: Vista Trabajadores

Situados nuevamente en el menú principal 7.3, al hacer clic sobre Servicios accedemos a la vista de la imagen 7.6. Verificamos que se cumple el requisito específico 3.10.

ID	Nombre	Foto	Instalación	Días Semana	Horario	Precio
8	Natación		Piscina	Lunes, Miércoles y Viernes	17.00-18.00	30
9	Gimnasio		Gimnasio	Lunes a Sábado	08.00-22.00	50
10	Yoga		Gimnasio	Martes y Jueves	08.00-09.00	40

Figura 7.6: Vista Servicios

Para cualquiera de las vistas anteriores tenemos la opción de búsqueda con filtros que nos ofrece Odoo. Podemos crear y guardar nuestros propios filtros, así como asignar favoritos.

Desde el listado de clientes 7.4 podemos pulsar sobre *Create* para crear un nuevo cliente, así como importar clientes mediante un CSV adjunto. Podemos observar la ventana de creación en la figura 7.7, cumpliendo así con el requisito específico 3.3.

SportService Clientes Trabajadores Servicios Administrator

Listado Clientes / New

Save Discard

Foto Perfil

ID

Nombre

Apellidos

Aniversario

Dirección

Email

Teléfono

Servicios

ID	Nombre	Foto	Instalación	Días Semana	Horario	Precio
Add a line						

Total Servicios 0

Pagado

Figura 7.7: Vista nuevo cliente

Al hacer clic sobre un cliente específico en el listado 7.4 accedemos a una vista específica para el cliente seleccionado 7.8. Aquí nos muestra toda la información ordenada, cumpliendo con el requisito específico 3.4. Si pulsamos en el botón *Edit* nos permite editar la información de dicho cliente, cubriendo así el requisito específico 3.5.

SportService Clientes Trabajadores Servicios Administrator

Listado Clientes / sportservice.cliente,3

Edit Create Action 1 / 3

**Foto Perfil**

**ID** 3

**Nombre** Pau

**Apellidos** Sastre Pons

**Aniversario** 05/01/1996

**Dirección** Pedreguer

**Email** pausastrepons@gmail.com

**Teléfono** 666666666

**Servicios**

ID	Nombre	Foto	Instalación	Días Semana	Horario	Precio
8	Natación		Piscina	Lunes, Miércoles y Viernes	17.00-18.00	30
9	Gimnasio		Gimnasio	Lunes a Sábado	08.00-22.00	50

**Total Servicios** 80

**Pagado**

Figura 7.8: Detalles cliente

Desde el listado de trabajadores 7.5 podemos pulsar sobre *Create* para crear un nuevo trabajador, así como importar trabajadores mediante un CSV adjunto. Podemos observar la ventana de creación en la figura 7.9, cumpliendo así con el requisito específico 3.7.

SportService Clientes Trabajadores Servicios Administrator

Listado Trabajadores / New

Save Discard

**Nombre**

**Apellidos**

**Número Seguridad Social**

**Aniversario**

**Teléfono**

**Foto Perfil**

**CV** Upload your file

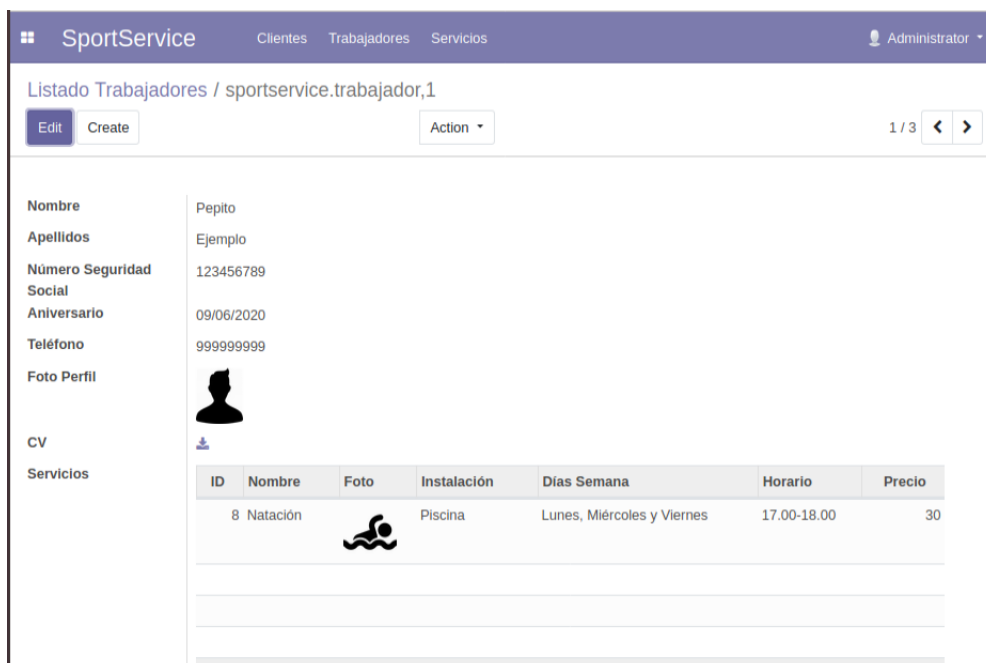
**Servicios**

ID	Nombre	Foto	Instalación	Días Semana	Horario	Precio
Add a line						

Figura 7.9: Vista nuevo trabajador

Al hacer clic sobre un trabajador específico en el listado 7.5 accedemos a una vista específica para el trabajador seleccionado 7.10. Aquí nos muestra toda la información

ordenada, cumpliendo con el requisito específico 3.8. Si pulsamos en el botón *Edit* nos permite editar la información de dicho trabajador, cubriendo así el requisito específico 3.9.



The screenshot displays the 'SportService' web application interface. The top navigation bar includes 'SportService', 'Clientes', 'Trabajadores', 'Servicios', and a user profile 'Administrator'. The main content area is titled 'Listado Trabajadores / sportservice.trabajador,1'. It features a sidebar with 'Edit' and 'Create' buttons, and a main section showing worker details: Nombre (Pepito), Apellidos (Ejemplo), Número Seguridad Social (123456789), Aniversario (09/06/2020), Teléfono (999999999), Foto Perfil (silhouette icon), and CV (download icon). Below this is a table of services provided by the worker.


ID	Nombre	Foto	Instalación	Días Semana	Horario	Precio
8	Natación		Piscina	Lunes, Miércoles y Viernes	17.00-18.00	30

Figura 7.10: Detalles trabajador

Desde el listado de servicios 7.6 podemos pulsar sobre *Create* para crear un nuevo servicio, así como importar servicios mediante un CSV adjunto. Podemos observar la ventana de creación en la figura 7.11, cumpliendo así con el requisito específico 3.11.



SportService Clientes Trabajadores Servicios Administrator

Listado Servicios / New

Save Discard

Foto

Nombre

Descripción

Instalación

Precio 0

Dias Semana

Horario

Aforo máximo 0

Aforo Actual 0%

Trabajadores

ID	Nombre	Apellidos	Foto Perfil
Add a line			

Clientes

ID	Nombre	Apellidos	Foto Perfil
Add a line			

Figura 7.11: Vista nuevo servicio

Al hacer clic sobre un servicio específico en el listado 7.6 accedemos a una vista específica para el servicio seleccionado 7.12. Aquí nos muestra toda la información ordenada, cumpliendo con el requisito específico 3.12. Si pulsamos en el botón *Edit* nos permite editar la información de dicho servicio, incluyendo la inscripción de clientes y trabajadores correctamente, cubriendo así el requisito específico 3.9.

The screenshot shows the 'SportService' application interface. The top navigation bar includes 'Cientes', 'Trabajadores', and 'Servicios', with the user 'Administrator' logged in. The main content area is titled 'Listado Servicios / sportservice.servicio,8'. Below this, there are buttons for 'Edit', 'Create', and 'Action', along with a pagination indicator '1 / 3'. The service details are as follows:

- Foto:** An icon of a swimmer.
- Nombre:** Natación
- Descripción:** Clases avanzadas de natación.
- Instalación:** Piscina
- Precio:** 30
- Días Semana:** Lunes, Miércoles y Viernes
- Horario:** 17.00-18.00
- Aforo máximo:** 30
- Aforo Actual:** 3%

Below the details, there are two tables: 'Trabajadores' and 'Clientes'.

ID	Nombre	Apellidos	Foto Perfil
1	Pepito	Ejemplo	[Profile Icon]

ID	Nombre	Apellidos	Foto Perfil
3	Pau	Sastre Pons	[Profile Icon]

Figura 7.12: Detalles servicio

La introducción de datos en todos los formularios es correcta, notificando de datos incorrectos y obligando a introducir datos obligatorios para el modelo. En la imagen 7.13 podemos ver como el sistema nos avisa de la introducción de un valor de aforo no válido. La figura 7.14 nos muestra una ventana de error al intentar superar el aforo máximo de un servicio.

The screenshot shows the 'SportService' application interface with an error notification dialog box overlaid. The dialog box is titled 'Número de 'aforo' incorrecto' and contains the message 'El número de aforo no puede ser negativo'. There is an 'Ok' button at the bottom of the dialog. The background shows the service details form with the following values:

- Nombre:** [Empty text field]
- Descripción:** [Empty text field]
- Instalación:** [Empty text field]
- Precio:** 0
- Días Semana:** [Dropdown menu]
- Horario:** [Dropdown menu]
- Aforo máximo:** -45
- Aforo Actual:** 0%

Figura 7.13: Notificación alta



Figura 7.14: Notificación alta

En las capturas de pantalla 7.15 y 7.16 observamos el correo automático que recibe el usuario Pau Sastre Pons al inscribirse y posteriormente darse de baja del servicio de gimnasio. Verificamos que se cumplen los requisitos específicos 3.14 y 3.15.

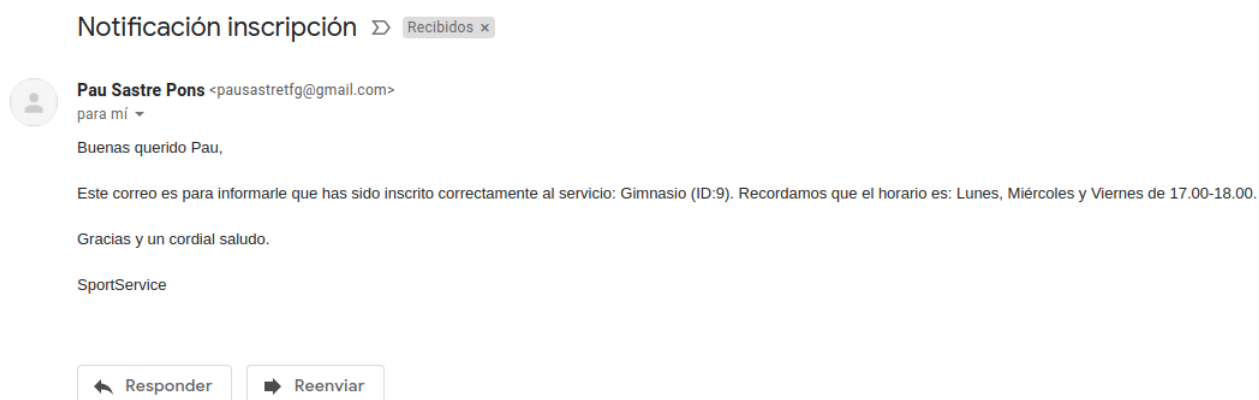


Figura 7.15: Notificación alta

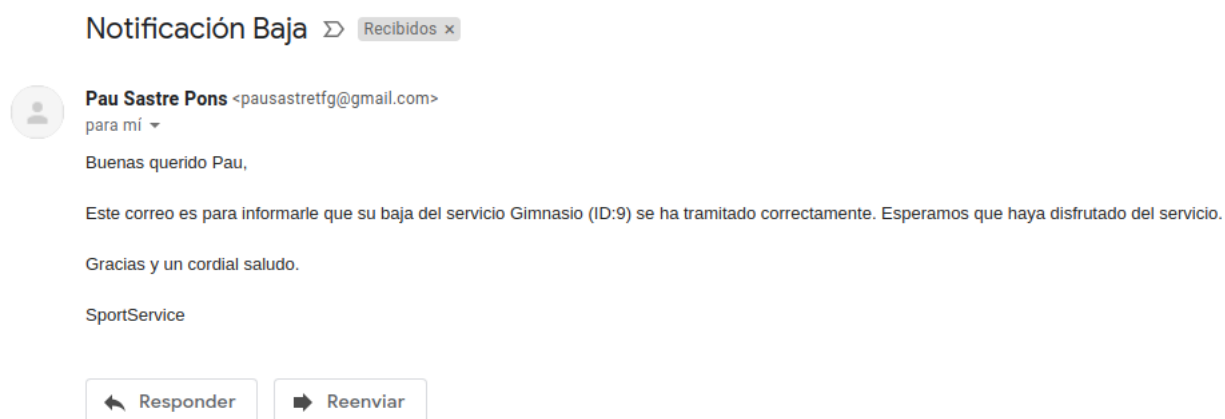


Figura 7.16: Notificación baja

Como hemos podido comprobar durante todo el apartado, la aplicación cumple satisfactoriamente con todos los requisitos planteados en el análisis 3. Se pueden llevar a cabo todos los casos de uso del apartado del diseño 4, y la interfaz de usuario se asemeja a los bocetos elaborados también en dicho apartado.

Por último comentar que existen otros tipos de pruebas no funcionales, por ejemplo pruebas de usabilidad, de estrés, de rendimiento, de escalabilidad, de portabilidad, etcétera. No se han realizado ninguna de estas, pero cabe destacar que el sistema ha arrancado y reiniciado correctamente durante las pruebas, sin ningún problema de inicio ni de rendimiento. En principio el sistema está configurado para ser escalable y portable, es uno de los objetivos de Odoo y así se ha configurado.

# Conclusiones y trabajos futuros

---

En este último apartado se realiza una valoración del proyecto una vez terminado, relacionando el resultado final con los objetivos planteados en el inicio. También se exponen los principales problemas surgidos durante el transcurso del proyecto. Por último, se enumeran una serie de posibles mejoras a implementar a la aplicación.

Los objetivos propuestos en el capítulo de Introducción 1 se han cumplido exitosamente. Al empezar el proyecto tenía una idea básica de que es un ERP, y al investigar para el proyecto he aprendido mucho sobre esta tecnología. Entre otras cosas, se han visto las principales alternativas del mercado actual, profundizando en Odoo, su historia, versiones y arquitectura. Como segundo objetivo cumplido se ha instalado y configurado exitosamente un sistema Odoo, completamente funcional y con una aplicación a medida personalizada. Para el desarrollo de dicha aplicación se han aplicado algunas de las metodologías de desarrollo software vistas durante el grado, cumpliendo así otro objetivo. Como último objetivo personal, se ha conseguido en su conjunto implementar un sistema de información útil y real.

Durante el desarrollo del proyecto, tanto la aplicación como la memoria, han surgido una serie de problemas. A continuación los contratiempos a destacar:

- **Aprender a utilizar LaTeX.** Al iniciar el proyecto sabía que era este procesador de textos, pero no lo había utilizado nunca. Supone un reto al ser mi primer trabajo en LaTeX, pero con la plantilla de la memoria y mis conocimientos previos de HTML me ha resultado bastante sencillo. La herramienta Overleaf <https://www.overleaf.com/> me ha ayudado mucho en la redacción, así como su documentación. Cabe destacar la dificultad de creación de tablas e insertar imágenes en LaTeX, pero resulta más fácil y práctico estructurar y referenciar el contenido.
- **Que es Odoo.** Como se ha comentado anteriormente, mis conceptos de un ERP eran muy básicos. Esto me ha ocupado mucho tiempo, investigando en distintas fuentes. Posteriormente he tenido que aprender sobre Odoo, tanto a nivel de conceptos como nivel técnico. Nunca había desarrollado en un software ERP. A nivel técnico, se ha instalado, desplegado y configurado el servicio, así como desarrollado un add-on. Durante la instalación surgieron problemas, sobretodo de librerías Python. Se consiguió solventar todo, mis conocimientos previos en sistemas Linux y Python fueron de gran ayuda.
- **Análisis y diseño.** Estos apartados de la memoria me supusieron un desafío, ya que no dominaba completamente la tecnología y no sabía que me podía ofrecer. Había puesto en practica las metodologías de desarrollo software utilizadas en proyectos más pequeños, en diversas asignaturas del grado, pero al principio me parecía que

no era apropiado para un ERP. Finalmente viendo las posibilidades de Odoo logré realizar un análisis y diseño correcto, que me ayudó mucho para el desarrollo posterior.

- **Uso del lenguaje Python.** En el grado no había cursado ninguna asignatura que utilizara este lenguaje, a pesar de su utilidad y popularidad. De manera independiente, yo había aprendido bastante Python para otros proyectos personales, básicamente para uso de administración de sistemas (*scripting*) y ciberseguridad (*exploiting*). En cambio dominaba bastante por los estudios cursados la programación orientada a objetos, con Java. Aprender POO con Python me supuso un esfuerzo considerable, ya que es muy diferente a Java en muchos ámbitos.

El proyecto cumple su propósito, pero durante su desarrollo se han detectado muchas mejoras potenciales a considerar de cara al futuro. Para implementar dichas mejoras se requiere más conocimientos de Odoo, librerías Python, hojas de estilo CSS, scripts en Javascript, y un largo etcétera. Por lo tanto, se requiere de más tiempo y estudio. Las principales mejoras detectadas son:

- **Mejora de la estética de la aplicación.** Mediante CSS y Javascript podemos editar nuestras vistas para que se muestren con otra estética, mejorando la distribución y el estilo del add-on.
- **Implementación de pagos clientes.** Mediante Paypal se puede implementar un sistema de cobro mensual automático a los clientes, se deberá introducir su cuenta bancaria o cuenta de Paypal. También se pueden generar facturas con Odoo, que se relacionarían con el pago mensual.
- **Portal web externo.** Como hemos comentado en el desarrollo, se puede implementar una página web externa con los datos que ofrece el módulo SportService como *backend* <https://www.odoo.com/documentation/12.0/howtos/website.html>. Esta mejora puede considerarse un proyecto aparte, ya que se trata de un desarrollo web añadiendo la tecnología Odoo. Las posibilidades son infinitas, podemos listar los servicios que se ofrecen, e incluso la opción de inscripción.
- **Mejora calendario.** Las propias herramientas de calendario que ofrece Odoo nos permiten editar y visualizar los atributos de horario de los servicios de forma más intuitiva y gráfica. También podemos representar horarios personalizados para cada cliente y trabajador.
- **Implementación nóminas trabajadores.** Odoo permite integrar las nóminas de los trabajadores, para ello hay muchos módulos ya desarrollados de recursos humanos y gestión de nóminas. Se trata de estudiar la mejor opción, o incluso desarrollar nosotros otro add-on o modelo.
- **Migración a Odoo 13.** La versión actual de Odoo es la 13.0, lanzada el 5 de octubre de 2019. El proyecto se inició con la versión 12.00 y no hubo necesidad de actualizarla. Hay que tener en cuenta que en octubre de 2020 sale la versión 14.00. En caso de decidir migrar hay que seguir los pasos del apartado 6.7, resolviendo las posibles complicaciones.
- **Aplicación para *smartphone*.** Desde la versión 10.0 de Odoo existe la opción de desarrollo de una app para teléfono móvil. Sólo esta disponible para la versión empresarial de Odoo, así que tendríamos que cambiar de licencia. Su desarrollo es similar al portal tradicional, añadiendo componentes nativos de teléfono móvil como son las notificaciones, vibración, cámara y gestos. Para más información: <https://www.odoo.com/documentation/12.0/reference/mobile.html>.

# Bibliografía

---

- [1] Enciclopedia libre online. <https://en.wikipedia.org/>
- [2] Daniel Reis. *Odoo 12 Development Essentials - Fourth Edition*. Packt Publishing Ltd, Birmingham, cuarta edición, 2018.
- [3] Informe ERP 2018, con múltiples datos estadísticos sobre el mercado en la actualidad llevado a cabo por la empresa Panorama Consulting Solutions. Consultado en <https://cdn2.hubspot.net/hubfs/2184246/2018%20ERP%20Report.pdf>
- [4] Documento en castellano sobre el ERS según IEEE 830. Disponible en el siguiente enlace <https://www.fdi.ucm.es/profesor/gmendez/docs/is0809/ieee830.pdf>
- [5] Federico Toledo Rodríguez. «Introducción a las pruebas funcionales». Introducción a las Pruebas de Sistemas de Información, 2014
- [6] Documentación oficial de Odoo. <https://www.odoo.com/documentation/12.0/>
- [7] Documentación oficial de Python. <https://www.python.org/doc/>
- [8] Documentación oficial de PostgreSQL <https://www.postgresql.org/docs/>





---

---

# APÉNDICE A

## Configuración del sistema

---

En este apéndice se explica la configuración para el desarrollo del proyecto. En primer lugar se explicará la máquina virtual y el sistema operativo, seguidamente la instalación y configuración de Odoo en la máquina y por último el IDE y otras herramientas de desarrollo utilizadas. Odoo también funciona en entornos Windows, en nuestro caso se ha optado por una máquina virtual linux.

### A.1 Máquina Virtual

---

Se ha decidido desarrollar sobre una máquina virtual sobre un sistema host linux Ubuntu 20.04. Se pretende simular un servidor linux para una empresa, con su entorno de desarrollo propio, por eso no utilizamos el sistema host, que tiene otros entornos y funcionalidades que podría interferir. Esta máquina virtual se podría portar fácilmente a otro sistema (linux, windows o mac). También instalar y configurar Odoo y sus dependencias en un sistema propio, o en un container (por ejemplo Docker o Kubernetes).

Para su instalación necesitamos una distribución basada en Debian, y hemos elegido Ubuntu por ser la distribución más estable y con más soporte para Odoo. Así que en primer lugar debemos descargar la última imagen ISO estable en su sitio web oficial: <https://ubuntu.com/download/desktop>. En nuestro caso es la versión 18.04, la mas estable cuando se inició este proyecto .

Para hospedar Ubuntu se ha utilizado VirtualBox, software de virtualización para arquitecturas x86/amd64 desarrollado por Oracle Corporation. Podemos descargarlo en su web oficial: <https://www.virtualbox.org/wiki/Downloads>. Tanto Ubuntu como VirtualBox cuentan con Licencia Pública General de GNU, el mismo tipo de licencia que Odoo CE.

Se ha realizado una instalación normal en la máquina virtual, con 8GB de memoria principal y 21GB de disco duro, más que suficiente para los requisitos mínimos de Ubuntu y Odoo. Se ha configurado el portapales bidireccional con la máquina host.

### A.2 Instalación y configuración de Odoo 12.0

---

Una vez puesto en marcha el sistema operativo debemos prepararlo para la instalación de Odoo. Para instalar Odoo se ha seguido la guía oficial: <https://www.odoo.com/documentation/master/setup/install.html#source-install>.

En primer lugar debemos actualizar e instalar los paquetes que lo necesiten con las siguientes ordenes del terminal:

```
1 $ sudo apt-get update
2 $ sudo apt-get dist-upgrade
```

Para que Odoo funcione necesitamos instalar PostgreSQL y crear un usuario:

```
1 $ sudo apt-get install postgresql-9.6
2 $ sudo su postgres
3 $ sudo adduser --system --home=/opt/odoo --group odoo
```

Añadimos el repositorio que proporciona Odoo S.A. con las siguientes líneas:

```
1 $ wget -O - https://nightly.odoo.com/odoo.key | apt-key add -
2 $ echo "deb http://nightly.odoo.com/12.0/nightly/deb/ ." >> /etc/apt/sources.
   list.d/odoo.list
3 $ apt-get update && apt-get install odoo
```

A continuación instalamos el gestor de paquetes Pip, para instalar y administrar los paquetes de Python. Mediante esta herramienta vamos a instalar todas las dependencias necesarias para lanzar Odoo. Un apunte importante, depende de cuando se instale puede haber cambios en las librerías o en el mismo Odoo, así que si a la hora de lanzar Odoo este no arranca porque le falta alguna librería o hay algún error en ella se puede resolver instalandola mediante Pip. A continuación la instalación de dependencias:

```
1 $ sudo apt-get install python3-pip
2 $ pip3 install babel decorator docutils ebaysdk feedparser gevent greenlet
   html2text Jinja2 lxml Mako MarkupSafe mock num2words ofxparse passlib
   Pillow psutil pycogreen pycopg2 pydot pyparsing PyPDF2 pyserial python-
   dateutil python-openid pytz pyusb PyYAML qrcode reportlab requests six suds
   -jurko vatnumber vobject werkzeug XlsxWriter xlwt xlrd
```

También instalamos el siguiente programa para detectar cambios en el código automáticamente:

```
1 $ sudo apt-get install python3-watchdog
```

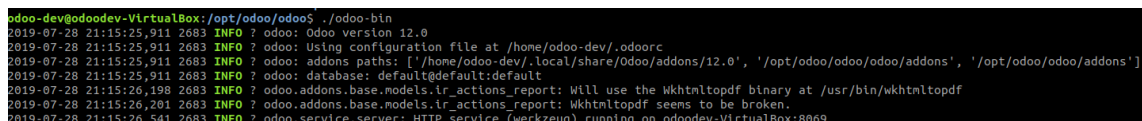
Instalamos Odoo CE desde su github oficial <https://github.com/odoo/odoo> mediante el comando git:

```
1 $ sudo apt-get install git
2 $ git clone https://github.com/odoo/odoo.git
```

Para comprobar si se ha instalado correctamente debemos arrancar Odoo y acceder a la siguiente dirección en nuestro navegador(en nuestro caso Firefox preinstalado de Ubuntu): <http://localhost:8069/web/database/selector>. En el enlace podemos ver el número de puerto por defecto 8069, con el que vamos a trabajar, se puede cambiar lanzando con la opción `-http-port="número nuevo"`.

```
1 $ cd /opt/odoo/odoo
2 $ ./odoo-bin
```

Si todo ha ido correctamente deberíamos ver en el navegador y el terminal lo que muestran las siguientes imágenes A.1 y A.2. En el navegador elegimos a que base de datos accedemos. Por defecto no nos pide contraseña, sin embargo se puede habilitar en los archivos de configuración.



```
odoo-dev@odoo-dev-VirtualBox: /opt/odoo/odoo$ ./odoo-bin
2019-07-28 21:15:25,911 2683 INFO ? odoo: Odoo version 12.0
2019-07-28 21:15:25,911 2683 INFO ? odoo: Using configuration file at /home/odoo-dev/.odoorc
2019-07-28 21:15:25,911 2683 INFO ? odoo: addons paths: ['/home/odoo-dev/.local/share/odoo/addons/12.0', '/opt/odoo/odoo/odoo/addons', '/opt/odoo/odoo/addons']
2019-07-28 21:15:25,911 2683 INFO ? odoo: database: default@default:default
2019-07-28 21:15:26,198 2683 INFO ? odoo.addons.base.models.ir_actions_report: Will use the Wkhtmltopdf binary at /usr/bin/wkhtmltopdf
2019-07-28 21:15:26,201 2683 INFO ? odoo.addons.base.models.ir_actions_report: Wkhtmltopdf seems to be broken.
2019-07-28 21:15:26,541 2683 INFO ? odoo.service.server: HTTP service (werkzeug) running on odoo-dev-VirtualBox:8069
```

Figura A.1: Terminal correcto

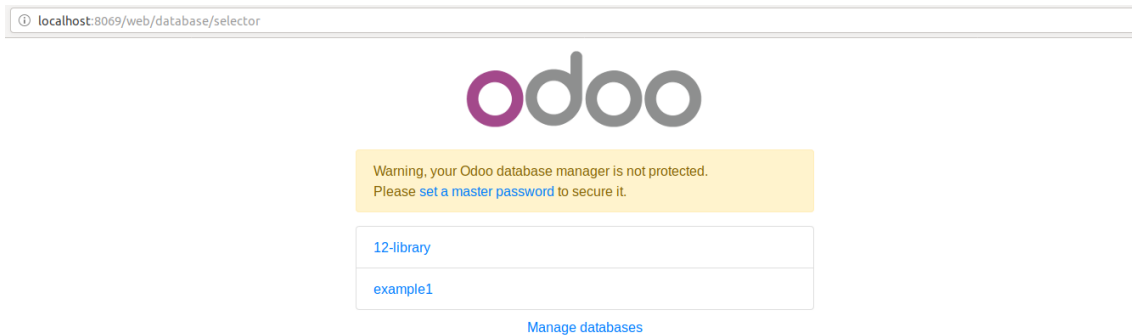


Figura A.2: Navegador correcto

Por último nos conviene modo desarrollador para hacer *debugging*, que carga los assets de uno en uno en lugar de compactarlos. Para activar este modo en el portal de Odoo necesitamos modificar la URL, y lo podemos hacer automáticamente mediante esta extensión de navegador: <https://addons.mozilla.org/en-US/firefox/addon/odoo-debug/>. Activamos y desactivamos pulsando el botón de la extensión como podemos observar en la imagen A.3.

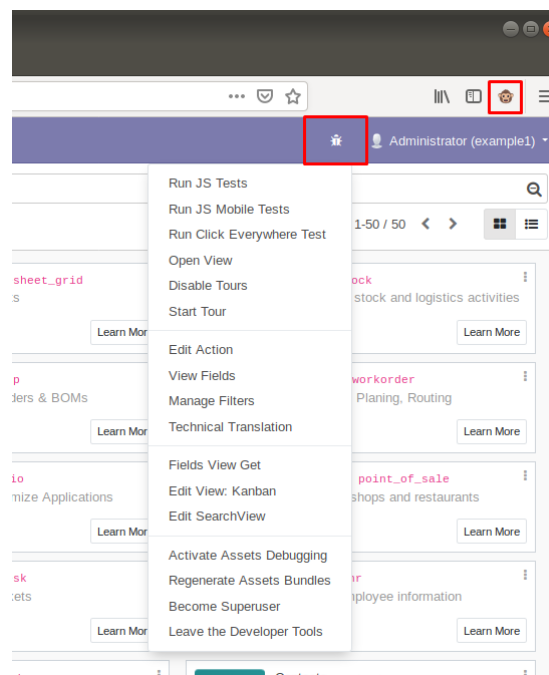


Figura A.3: Modo desarrollado

## A.3 Otras herramientas

Aquí se explican brevemente otro software utilizado en el desarrollo.

### A.3.1. Terminator

Terminator es un emulador de terminal basado en el terminal de GNOME, te permite utilizar múltiples pestañas y paneles redimensionables en una sola ventana. Facilita y simplifica el uso de muchos terminales a la vez, sobretodo si necesitas verlos todos simultáneamente. Se puede instalar a través de la aplicación Ubuntu Software o con la siguiente línea de comandos:

```
$ sudo apt-get install pycharm-community
```

### A.3.2. PyCharm

PyCharm es un entorno de desarrollo integrado, *Integrated Development Environment* (IDE) en inglés, que facilita el desarrollo de código Python, así como organizar archivos, depurar código, y muchas cosas más. Utilizamos la *Community Edition*, que es la versión libre bajo la licencia Apache 2.0 License. Como anteriormente, se puede instalar a través de la aplicación Ubuntu Software o con la siguiente línea de comandos:

```
$ sudo apt-get install terminator
```

### A.3.3. Pencil

Pencil es una aplicación de escritorio multiplataforma para prototipado, en este caso usado para realizar los mockups de la interfaz de usuario. Es gratuito y de código libre, bajo la licencia GNU LGPL v2. Se ha utilizado la última versión para Ubuntu disponible en la web oficial: <https://pencil.evolus.vn/>.

### A.3.4. Lucidchart

Herramienta web que nos permite realizar toda clase de diagramas en el navegador de manera intuitiva, con una interfaz fácil y amigable. Es muy útil para todo tipo de diagramas relacionados con el diseño de software. En este proyecto se ha utilizado para realizar los diagramas de caso de uso y para el diagrama de clases, siguiendo el formato UML. La licencia gratuita es suficiente para este proyecto, ya que se trata de un proyecto individual y no requiere de muchos documentos o otras características que ofrecen las versiones de pago. Su enlace es: <https://app.lucidchart.com>.

---

---

## APÉNDICE B

# Acrónimos

---

<b>ERP</b>	<i>Enterprise Resource Planning</i>
<b>MRP</b>	<i>Material Requeriment Planning</i>
<b>MRP II</b>	<i>Manufacturing Requeriment Planning</i>
<b>CRM</b>	<i>Customer Relationship Management</i>
<b>ORM</b>	<i>Object-Relational Mapping</i>
<b>SaaS</b>	<i>Software as a Service</i>
<b>IaaS</b>	<i>Infrastructure as a Service</i>
<b>PaaS</b>	<i>Platform as a Service</i>
<b>PYMES</b>	Pequeña Y Mediana Empresa
<b>VoIP</b>	<i>Voice Over IP</i>
<b>W3C</b>	<i>World Wide Web Consortium</i>
<b>IDE</b>	<i>Integrated Developement Environment</i>
<b>ERS</b>	Especificación de Requisitos Software
<b>MVC</b>	Modelo Vista Controlador
<b>UML</b>	<i>Unified Modeling Language</i>
<b>XML</b>	<i>eXtensible Markup Language</i>
<b>CSS</b>	<i>Cascading Style Sheets</i>
<b>CSV</b>	<i>Comma-separated Values</i>
<b>ACL</b>	<i>Access-Control List</i>
<b>POO</b>	Programación Orientada a Objetos