Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Desarrollo de un módulo del ERP Odoo para la inscripción de estudiantes a las clases en una academia

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Alejandro López Galiana

Tutor: José Vicente Busquets Mataix

Curso 2020-2021

Resumen

Este proyecto va a consistir en un trabajo conjunto sobre el desarrollo de una aplicación cuyo objetivo va a ser el de proporcionar la gestión y la planificación de una academia de cualquier tipo. Para esto se va a utilizar el ERP Odoo, ya que este se adapta bien a las necesidades planteadas para nuestro problema. Se realizará este proyecto puesto que existen muchas academias las cuales aún no cuentan con un sistema de gestión informatizado y se ha considerado que este puede ser de gran utilidad a cualquier academia que lo desee, independientemente de la materia que imparta ya que la aplicación se puede adaptar a cualquier escenario.

Este trabajo se ha dividido en tres módulos de las cuales mis compañeros se van a encargar de la planificación de las clases y de la asistencia de los alumnos, y yo de la gestión de los usuarios, mediante su creación y eliminación de la academia.

Como se va a ver la aplicación se desarrollará exitosamente y se obtendrán los resultados esperados creando un módulo funcional el cual ejecuta las principales funciones que requiere una academia.

Palabras clave: ERP, Odoo, Módulo, academia

Resum

Aquest projecte consistirà en un treball conjunt sobre el desenvolupament d'una aplicació l'objectiu de la qual serà el de proporcionar la gestió i la planificació d'una acadèmia de qualsevol tipus. Per a això s'utilitzarà el ERP Odoo, ja que aquest s'adapta bé a les necessitats plantejades per al nostre problema. Es realitzarà aquest projecte perquè existeixen moltes acadèmies les quals encara no compten amb un sistema de gestió informatitzat i s'ha considerat que aquest pot ser de gran utilitat a qualsevol acadèmia que ho desitge, independentment de la matèria que impartisca ja que l'aplicació es pot adaptar a qualsevol escenari.

Aquest treball s'ha dividit en tres mòduls de les quals els meus companys s'encarregaran de la planificació de les classes i de l'assistència dels alumnes, i jo de la gestió dels usuaris, mitjançant la seua creació i eliminació de l'acadèmia.

Com es veurà l'aplicació es desenvoluparà reeixidament i s'obtindran els resultats esperats creant un mòdul funcional el qual executa les principals funcions que requereix una acadèmia.

Paralules clau: ERP, Odoo, Mòdul, acadèmia

Abstract

This project will consist of a joint work on the development of an application whose objective will be to provide the management and planning of an academy of any kind. For this, the Odoo ERP will be used, since it adapts well to the needs raised for our problem. This project will be carried out since there are many academies which not have a computerized management system and it has been considered that this can be very useful to any academy that wishes, regardless of the subject that it teaches since the application can be adapted to any setting.

This work has been divided into three modules of which my colleagues are going to be in charge of planning the classes and the attendance of the students, and I of the management of the users, through their creation and elimination of the academy.

As you are going to see, the application will be developed successfully, and the expected results will be obtained by creating a functional module which executes the main functions that an academy requires.

Keywords: EPR, Odoo, module, academy

Índice Gerenal

1. Introducción 11
1.1 Motivación 11
1.2 Objetivos 11
1.3 Estructura 11
2. Colaboraciones 14
3. Estado del arte 16
3.1 ERP 16
3.1.1 Historia del ERP 16
3.1.2 Características de un ERP 17
3.1.3 ERP en las empresas actualmente 17
3.2 Odoo 21
3.2.1 Historia de Odoo 21
3.2.2 características Odoo 22
3.2.3 Arquitectura de Odoo 22
4. Análisis 24
4.1 Introducción 24
4.2 Descripción general 24
4.3 Requisitos específicos: 26
4.4 Ejemplo de uso concreto 30
4.5 Diagramas de casos de uso 31
4.6 Prototipado 33
5. Diseño de la solución 37
5.1 Diagrama de clases 37
6. Desarrollo de la solución 39
6.1 Modelos 40
6.2 wizards 42
6.3 Vistas 44
6.4 Seguridad 48
6.5 Otros ficheros 48

7. Pruebas funcionales. 51
8. Conclusiones y proyectos futuros. 57
9. BIBLIOGRAFIA 60
Apéndice 1: Configuración del sistema 63
A.1 Máquina Virtual 63
A.2 Instalación de Odoo 13.0 63
A.3 Instalacion Atom 64

Índice de figuras

Figura 1: gráfico de la selección de decisiones según panorama consulting 2021 18
Figura 2 : Gráfico de la dificultad de los trabajadores en el cambio de un ERP 20
Figura 3 Arquitectura Odoo 23
Figura 4: Diagrama de casos de uso alumno.....31
Figura 5: Diagrama de casos de uso profesor 32
Figura 6: Diagrama de casos de uso administrador. 33
Figura 7: Prototipo lista usuarios 34
Figura 8: Prototipo introducir datos usuarios 34
Figura 9: Prototipo suscribir alumno 35
Figura 10: Diagrama de clases.....37
Figura 11: Estructura clases 39
Figura 12: inicio sesión Odoo 51
Figura 13: Menu Odoo 52
Figura 14: Lista usuarios.....52
Figura 15: Inscripción usuario 52
Figura 16: Calendario clases filtro 53
Figura 17: Calendario clases no filtro.....53
Figura 18: Información clase 54
Figura 19: Suscripción clase 54
Figura 20: Información clase alumno suscrito 54
Figura 21: información clase 55

Índice de tablas

Tabla 1: Evolución de ERP en empresas.....21
Tabla 2: Evolución de Odoo 22
Tabla 3: Funciones equipo 25
Tabla 4: Restricción 1 26
Tabla 5: Restricción 2 27
Tabla 6: Restricción 3 27
Tabla 7: Restricción 4 27
Tabla 8: Restricción 5 28
Tabla 9: Restricción 6 28

Tabla 10: Restricción 7 28
Tabla 11: Restricción 8 29
Tabla 12: Restricción 9 29
Tabla 13: Restricción 10 29
Tabla 14: Restricción 11 30
Tabla 15: Restricción 12 30

1. Introducción

Comenzando por la introducción esta va a ser dividida en tres partes: motivación, objetivos y estructura de la memoria. La motivación se centra en por qué ve a ser desarrollado este proyecto, el objetivo va a hablar del fin de la aplicación desarrollada y por último la estructura va a explicar los capítulos de los que esta va a estar compuesta.

1.1 Motivación

Hoy en día la gestión de la información es una de las partes más importantes de cualquier empresa y la tecnología más predominante en este aspecto son los ERP, por tanto, veo de vital importancia aprender el uso y funcionamiento de estos para mi futuro laboral, otro aspecto a considerar es el hecho de que muchas pequeñas empresas en el sector, (sobre todo pymes), no tienen implementado un software de este tipo, y les sería de gran ayuda, en muchos de sus aspectos.

Además, tengo un especial interés en desarrollar este TFG en con el ERP Odoo, ya que, utiliza software libre, con el cual cualquier persona tiene libertad para modificarlo y ver lo que otros usuarios realizan en este con el fin de mejorar el ERP de manera conjunta.

1.2 Objetivos

El objetivo principal de este TFG va a ser el de desarrollar una aplicación mediante el uso de Odoo. Principalmente esta aplicación va a ser tratada sobre una academia de música, en la cual los alumnos podrán suscribirse apuntarse a las clases deseadas, y se llevará registro de las asistencias de estos. Como objetivos personales podrían enumerarse el de Entender que es un ERP, Odoo y su utilidad, la correcta instalación del sistema Odoo y desarrollar una aplicación real combinando todo lo anterior y todo lo aprendido durante el grado de ingeniería informática.

1.3 Estructura

Para finalizar la introducción voy a describir las partes en las que se va a dividir la memoria del proyecto.

- Introducción: Apartado en el que nos hallamos, explica la motivación, los objetivos y la estructura del proyecto.
- Colaboraciones: Nombra a los distintos colaboradores que he tenido durante este TFG y de que parte se encarga cada uno.
- Estado del arte: Apartado que va a explicar detalladamente que es un ERP, su historia, sus características y su desempeño en las empresas actualmente, también va a explicar que es Odoo, su historia y arquitectura.
- Análisis: Planteamiento del caso de estudio usando la especificación de requisitos de software que marca el estándar IEEE 830.

- Diseño de la solución: Planteamiento de la solución del problema mediante técnicas como casos de uso, diagrama de clases o prototipos.
- Desarrollo de la solución: Amplio análisis de las clases que han compuesto nuestro proyecto.
- Implantación y mantenimiento: Despliegue y configuración de Odoo sobre Ubuntu.
- Pruebas funcionales: Pruebas que se realizan para comprobar el correcto funcionamiento del módulo desarrollado.
- Conclusiones y trabajos futuros: Finalización del proyecto que se va a centra en comprobar si los objetivos han sido cumplidos, y en cómo mejorar la aplicación.
- Apéndice: Configuración de tanto Odoo, como de los programas necesarios para realizar el proyecto.

2. Colaboraciones

Aquí vamos a ver a los miembros del equipo que van a realizar la aplicación de la academia, esto se debe a que cada uno de nosotros va a realizar un módulo, el cual por sí solo no otorga las funcionalidades que una academia necesitaría y cuando estos estén acabados vamos a juntarlos para crear una aplicación totalmente funcional.

Para empezar los miembros del equipo son Jinye Ji, Sergio Galbis, Diego Sahuquillo y yo. El plan es que yo me encargue del tema de inscripción de alumnos y de profesores tanto a la academia como a las clases, así como relacionar la base de datos de Odoo con nuestra lista de usuarios, de tal forma que una vez se registra un usuario en la academia, también se crea un nuevo usuario en Odoo, y este puede iniciar sesión en esta. Jinye se encarga de la planificación de las clases, creándolas, asignándoles horarios y realizando la planificación necesaria. Sergio va a realizar principalmente de la asistencia a las clases, con funciones como llevar registros de la asistencia de los alumnos, o fijar límites de alumnos para las clases puesto que las clases deberán tener un límite de alumnos que puedan apuntarse a esta. Finalmente, Diego se va a encargar del tema económico de la academia, mediante acciones como fijar el precio a las clases o generar las facturas necesarias para pagar las clases.

Debido a este reparto de tareas mi parte está estrechamente relacionada con la parte de Jinye, puesto que yo suscribo alumnos a las clases y él crea las clases a las que se suscriben los alumnos, por lo tanto, en ocasiones se necesitará heredar parte de su código para que mi módulo pueda funcionar, así mismo tanto Sergio como Diego necesitarán en alguna ocasión parte de mi código para por ejemplo para consultar atributos de los alumnos como su identificador o su correo.

Importante remarcar antes de acabar ente punto que Diego realizará su modulo después que el resto de los integrantes que lo estamos realizando al mismo tiempo, por lo que sus funcionalidades todavía no están seguras completamente y es la que más sujeta a cambios están.

3. Estado del arte

Una vez hemos acabado la introducción, es el momento de hablar del estado del arte, en este apartado se va a explicar tanto que es un ERP, como la importancia de este en la actualidad, así como su historia y los diferentes tipos de ERP que existen y las características principales de Odoo, así como una descripción de su arquitectura.

3.1 ERP

Para comenzar vamos a hablar de la historia del ERP, tanto de cómo se creó, como de su evolución hasta la época actual.

3.1.1 Historia del ERP

A finales de 1940 y principios de 1950, al igual que muchos avances, debido a, la segunda guerra mundial el ejército de Estados Unidos desarrolló programas informáticos para poder gestionar las tareas de producción y logística del esfuerzo bélico, aunque estos programas solo se aplicaron al ejército y era muy primitivo estos se consideran el origen de los futuros ERP.

En 1960, a raíz de las primeras computadoras para empresas se produje un gran avance a la hora de la gestión de la información, en aquella época lo más común era la entrega del software básico con la compra del hardware, aunque después de esto era posible la contratación de desarrollos para adaptar las necesidades de las compañías. Así llegó a las empresas las aplicaciones básicas BOM (Listas de Materiales, en inglés), las cuales se encargaban de adaptar las herramientas de planificación que fueron desarrolladas por el ejercito la anterior década. Durante esta época, fue patente que vender software podría ser altamente beneficioso para una compañía y surgieron las primeras empresas dedicadas únicamente al desarrollo de software

En la época de 1970 aparecieron programas los cuales se empezaban a asemejar a lo que ahora conocemos como ERP, los MRP (Planificación de Necesidades de Materiales), los cuales en contraposición a aplicaciones desarrolladas anteriormente podían controlar, tanto el uso de los materiales, como poder prever cuando iban a necesitarlos y que cantidad. Además, durante esta época se fundaron proveedores de programas ERP muy conocidos como SAP (1972) o Baan (1978).

En 1980 los programas usados por las compañías para la planificación de la producción evolucionaron más con el fin de poder incluir otros aspectos a parte del de las materias primas. Sus nombres cambiaron al de MRP-II (Planificación de Recursos de Producción) y se empezaron a introducir aspectos financieros como el coste de obtención de las materias primas, el coste de mano de obra… Gracias a los MRP nacieron compañías que más adelante se especializarían en el desarrollo de ERP.

Por fin en 1990 nace el ERP, se le atribuye a la consultora Gartner la acuñación del término, la principal razón por la que se dejó de lado el termino MRP es debido a que los ERP eran capaces de definir nuevos programas de planificación empresarial, cuyo alcance podía

superar los ámbitos tradicionales. Además de esto, no estaba restringido solo a software de compañías de fabricación, sino que era capaz de ser utilizado por empresas de cualquier tipo.

Para finalizar en los 2000 los ERP se popularizaron y empezaron a integrar funciones que hasta entonces realizaban otras aplicaciones, como la gestión de las relaciones con los clientes (CRM) o la gestión de la cadena de suministro (SCM) [1].

3.1.2 Características de un ERP

Una vez explicada la historia de los ERP voy a hablar de sus principales características.

- *'Un sistema integrado.*
- *Opera en (o casi) tiempo real.*
- *Una base de datos común que admite todas las aplicaciones.*
- *Una apariencia uniforme en todos los módulos.*
- *Instalación del sistema con una elaborada integración de aplicaciones / datos por parte del departamento de Tecnología de la Información (TI), siempre que la implementación no se realice en pequeños pasos.*
- *Las opciones de implementación incluyen: local, alojado en la nube o SaaS.'*[2]

3.1.3 ERP en las empresas actualmente

A lo largo de los años las empresas van variando sus preferencias a la hora de escoger un ERP, pero una cosa permanece constante, las organizaciones todavía tienen un número cada vez mayor de sistemas ERP para escoger. Muchos de estos sistemas ERP brindan una amplia gama de capacidades y tienen las mejores prácticas integradas de la industria. Con tantos productos funcionales en el mercado, las organizaciones a menudo realizan pruebas para los ERP, y estos acaban clasificados en los siguientes niveles:

Nivel 1: Estos sistemas están diseñados para empresas con más de 750 millones de dólares en ingresos anuales. La mayoría de las empresas de este tamaño son complejas, ya sea por procesos operativos complejos o por la complejidad de la estructura de la entidad y sus necesidades de consolidación. Las aplicaciones de Nivel 1 abordan múltiples industrias y escalabilidad.

Ejemplos: SAP S/4HANA, Oracle ERP Cloud, the Infor CloudSuites

Nivel 2 alto: Estos sistemas suelen servir a organizaciones pequeñas y medianas con entre 250 y 750 millones de dólares en ingresos anuales, las organizaciones de este tamaño pueden abarcar múltiples industrias y múltiples unidades de negocios.

Ejemplos: Microsoft Dynamics 365 Finance, IFS, Sage X3, Epicor E10 [16].

Nivel 2 bajo: Estos sistemas generalmente sirven a organizaciones pequeñas y medianas con 10 millones a 250 millones de dólares en ingresos anuales, estas organizaciones generalmente representan solo una industria y tienen una sola entidad para administrar.

Ejemplos: NetSuite, abas, IQMS, Plex, Microsoft Dynamics 365 Business Central, SYSPRO, Acumatica[16].

Nivel 3: Hay cientos de proveedores de software en este nivel que prestan servicios en su mayoría a organizaciones más pequeñas, sin embargo, también existen algunas soluciones puntuales muy robustas con funcionalidad de nicho que a menudo se utilizan para complementar un sistema ERP más grande.

Ejemplos: Sage ERP 100, Sage ERP 300, Aptean, ECI, ASC.[16]

Al seleccionar un sistema ERP para los objetivos comerciales actuales y futuros, es importante que las organizaciones consideren qué funcionalidad necesitarán dentro de los próximos cinco años. Una vez que una organización entiende sus objetivos, puede determinar si un proveedor en particular encajaría bien en el largo plazo, las organizaciones a menudo buscan grandes proveedores de ERP, como SAP, Oracle y otros proveedores de Nivel I. La ventaja que tienen los grandes proveedores sobre los más pequeños son las aplicaciones adicionales en su cartera para complementar su sistema ERP principal. Sin embargo, el Nivel I no es de ninguna manera la categoría para encontrar el mejor software ERP para cada organización A continuación se muestra una descripción general de la variedad de decisiones de selección de las organizaciones [3].

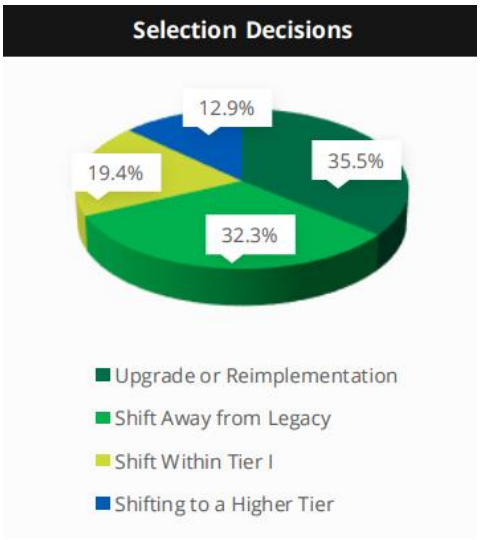

Figura 1: gráfico de la selección de decisiones según panorama consulting 2021

-Actualización o reimplementación: muchas organizaciones deciden mantener su sistema ERP actual y actualizarlo o reimplementarlo. En otras palabras, el sistema que indicaron que estaban reemplazando era el mismo sistema que estaban utilizando. En comparación con el año pasado, hubo un aumento del 14 en las organizaciones que conservaron su sistema, muchas organizaciones que tienen la intención de actualizar su sistema terminan reimplementándolo en su lugar. Por ejemplo, si una organización tiene un sistema ERP altamente personalizado, la actualización puede ser esencialmente una reimplementación A pesar de los desafíos de la actualización, parece que muchas organizaciones deciden mantener su software actual.

-Alejándose del software heredado: En el competitivo entorno empresarial actual, muchas organizaciones descubren que no tienen más remedio que implementar un sistema moderno si quieren seguir el ritmo de sus competidores. Los sistemas heredados no pueden

integrar y analizar grandes cantidades de datos o proporcionar información de datos en tiempo real. Los datos en tiempo real son especialmente importantes cuando se trata de mejorar la experiencia del cliente u optimizar la cadena de suministro, por ejemplo, cuando el inventario se administra a través de un sistema ERP moderno con datos en tiempo real, los datos del inventario son mucho más precisos de lo que serían en un sistema heredado. Esto se debe a que los sistemas ERP modernos son sistemas totalmente integrados con una sólida conectividad, estos sistemas tienen capacidades avanzadas que brindan análisis procesables impulsados por inteligencia artificial y aprendizaje automático,

-Cambiar a otro nivel 1: Este es un escenario relativamente poco común entre las empresas, lo cual no es sorprendente ya que las organizaciones necesitarían una razón comercial muy convincente para cambiar a un sistema que sea bastante comparable a su sistema existente. Los sistemas de nivel 1 son comparables en el sentido de que todos son altamente escalables y adecuados para una variedad de industrias.

-Cambiar a un nivel superior: Solo el 13% de las organizaciones se trasladaron a un sistema ERP de nivel superior. Este fue el escenario menos común, lo que podría deberse al hecho de que las capacidades de los sistemas ERP nivel 2 y nivel 3 son ahora más avanzadas que nunca. Esto significa que las organizaciones tienen menos motivos para cambiar a un sistema de Nivel 1, ya que ya tienen un sistema con la profundidad adecuada de funcionalidad y escalabilidad. Incluso si implementaron estos sistemas hace muchos años, estos pueden ser sistemas ERP en la nube que el proveedor ha estado actualizando automáticamente de forma regular.[15]

Las empresas tienden a centrar la mayor parte de su atención en los aspectos técnicos de los proyectos, sin embargo, para lograr los beneficios comerciales esperados, las organizaciones deben centrarse tanto, si no más, en los aspectos de las personas y los procesos. Por ejemplo, antes de seleccionar el software ERP es importante definir una estrategia digital y buscar oportunidades de mejora dentro de los procesos del estado actual. También es fundamental comunicarse con los empleados antes de la selección para que se sientan involucrados en el proyecto y puedan prepararse mentalmente para adoptar nuevos procesos y tecnología. En el siguiente gráfico se muestra la dificultad de los trabajadores a la hora de ejecutar los cambios en un ERP separando por dificultad en el cambio de proceso, en el cambio organizacional y el cambio técnico.

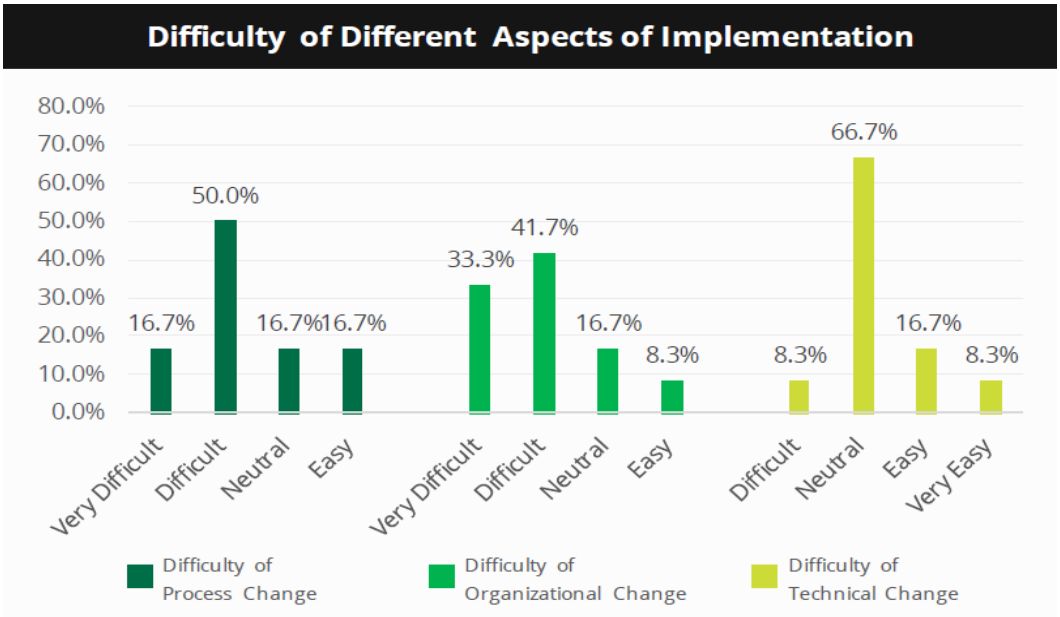

Figura 2 : Gráfico de la dificultad de los trabajadores en el cambio de un ERP

Aquí se puede ver como los cambios de proceso y los cambios organizacionales son considerados más difíciles por una gran cantidad de trabajadores a diferencia de los cambios técnicos los cuales son fácilmente abordables según los trabajadores.

Para finalizar de hablar de los ERP voy a mostrar una tabla la cual muestra la diferencia de los ERP a lo largo de los últimos años según los informes redactados por panorama consulting.[2]

	Informe ERP 2018	Informe ERP 2019	Informe ERP 2020	Informe ERP 2021
Industrias principales	Fabricación y distribución	Fabricación e información tecnológica	Fabricación e información tecnológica	Fabricación e información tecnológica
Ingresos anuales	439 millones $	1.78 billones$	1.98 billones $	28.4 millones$
Enfoque BPM	49% mejoró procesos	30% mejoró procesos	37% mejoró procesos	50% mejoró procesos
Fidelidad del presupuesto	36% se mantuvo dentro del presupuesto	55% se mantuvo dentro del presupuesto	62% se mantuvo dentro del presupuesto	40% se mantuvo dentro del presupuesto
Razón principal para Exceso de presupuesto	Problemas técnicos no anticipados	Expansión del alcance	Expansión del alcance	Tecnología adicional
Cumplimiento de horarios	21% cumplió los horarios	42% cumplió los horarios	53% cumplió los horarios	54% cumplió los horarios

Razón principal para Exceso de horarios	Cuestiones de organización	Cuestiones de organización	Cuestiones de organización	Cuestiones de organización

Tabla 1: Evolución de ERP en empresas

Se puede observar que cuando el ingreso anual promedio fue inferior a 1 B, los encuestados informaron un mayor enfoque en la gestión de procesos de negocio, Sin embargo, esto fue acompañado por una menor capacidad para mantenerse dentro del presupuesto. Cuando el ingreso anual promedio fue más de 1 B, los encuestados tenían más probabilidades de expansión. Los problemas organizacionales han plagado constantemente a las organizaciones a lo largo de los años.

Como conclusión de este tema, se puede destacar que un fuerte enfoque en la gestión del cambio es fundamental para las empresas, ya sea que una organización esté realizando una actualización de ERP, una implementación de ERP o una transformación digital. Centrarse en la gestión del cambio organizacional y buscar orientación relacionada puede mitigar la dificultad del cambio organizacional, pero menos organizaciones se centran en las actividades de gestión del cambio.

Otro punto a tener en cuenta es que los beneficios aumentaron todos los años excepto en 2021, esto podría deberse a que las organizaciones están ajustando sus expectativas en función de la pandemia, aunque estos datos no reflejan todo el año y aún puede subir. No obstante, no es poco realista suponer que muchas de estas organizaciones todavía están estableciendo objetivos lo suficientemente ambiciosos para al menos recuperar el costo de sus proyectos de ERP.

3.2 Odoo

En este apartado voy a hablar de Odoo, me voy a centrar en la historia, sus características, sus ediciones y su arquitectura.

3.2.1 Historia de Odoo

En 2005 Fabien Pinckaers fundador y actual CEO de Odoo empezó a desarrollar un ERP llamado TinyERP para intentar plantar cara a otros grandes ERP como SAP, un año más tarde Fabien compró el dominio SorrySAP.com para conseguir “derrotar” a SAP como el ERP número uno, durante los próximos tres años la compañía creció exponencialmente hasta que en 2010 cambiaron el nombre de TinyERP por el de OpenERP(debido en gran parte a que las grandes compañías no querían un software que “fuese enano”) y se consolido como compañía teniendo más de 100 empleados. Los siguientes años fueron años de expansión tanto económica como territorialmente hasta el punto de que en 2014 se daban alrededor de 1000 instalaciones por día convirtiéndose ese año en el software de gestión más grande del mundo, lanzando alrededor de 60 nuevos módulos al mes y con una red de usuarios mayor de dos millones. Para finalizar en 2014, la empresa pasó a llamarse Odoo, para diversificarse del término ERP y en 2015, Inc. Magazine colocó a Odoo entre las 5000 empresas privadas de más rápido crecimiento en Europa [4].

A lo largo de los años se fueron dando versiones de Odoo con diferentes nombres como se puede ver en la siguiente tabla:

Nombre programa	Versión	Fecha lanzamiento	Tipo software
TinyErp	1.0	Febrero 2005	GNU GPL
	2.0	Mayo 2005	GNU GPL
	3.0	Septiembre 2005	GNU GPL
	4.0	Diciembre 2006	GNU GPL
OpenERP	5.0	Abril 2009	GNU GPL
	6.0	Enero 2011	GNU GPL
	6.1	Febrero 2012	GNU GPL
	7.0	Diciembre 2012	GNU GPL
Odoo	8.0	Septiembre 2014	GNU GPL
	9.0	Octubre 2015	GNU GPL v3/ Odoo Enterprise Edition License v1.0
	10.0	Octubre 2016	GNU GPL v3/ Odoo Enterprise Edition License v1.0
	11.0	Octubre 2017	GNU GPL v3/ Odoo Enterprise Edition License v1.0
	12.0	Octubre 2018	GNU GPL v3/ Odoo Enterprise Edition License v1.0
	13.0	Octubre 2019	GNU GPL v3/ Odoo Enterprise Edition License v1.0

Tabla 2: Evolución de Odoo

3.2.2 características Odoo

Odoo al ser un ERP cuenta con las características ya mencionadas anteriormente, pero además de esto cuenta con otra serie de características que la hacen tan única y distinta a muchas de sus alternativas.

Para empezar Odoo es de código abierto, esto provee una libertad que prácticamente ningún otro ERP puede igualar, esto propicia que puedas modificar códigos ya creados e implementarlos e tus sistemas u obtener solo las partes que necesites, además no es necesario ningún proveedor para el funcionamiento de Odoo, por lo tanto, puedes elegir el distribuidor que más te convenga para tu proyecto. También cabe destacar la conectividad que ofrece con otros servicios software gracias al webservice, por el cual permite la visualización de archivos PDF o la importación/exportación de archivos CSV. Además, y aunque no sea una característica intrínseca Odoo ya es un ERP bastante popular proporcionándole un gran número de módulos ya creados por otros usuarios que pueden resultar de gran ayuda, o incluso foros con usuarios en los que se resuelven una gran variedad de dudas.[8]

Para finalizar hay que comentar la tecnología que uso siendo PostgreSQL su unidad de base de datos, Python el lenguaje en el que se programa y XML (eXtensible Markup Language) el lenguaje que se usa para crear las interfaces.

3.2.3 Arquitectura de Odoo

A continuación, se explican las distintas partes que componen la arquitectura de Odoo

Capa de datos.

Esta es la capa más baja de las tres, se dedica a almacenar y otorga persistencia a los datos usando para ello PostgreSQL. Además de esto, la capa de datos da información a la capa lógica, la cual es la superior a la de datos.

Capa Lógica.

La capa intermedia de las tres, y la que contiene los programas que van a ser ejecutados, además de recibir y enviar las peticiones del usuario. También se encarga de relacionar la capa de datos y la de presentación entre ellas.

Capa de presentación.

Para finalizar, la capa de más nivel, y la que se encarga de mostrar la interfaz por pantalla, por tanto, es la que el usuario ve cuando se conecta a Odoo [6].

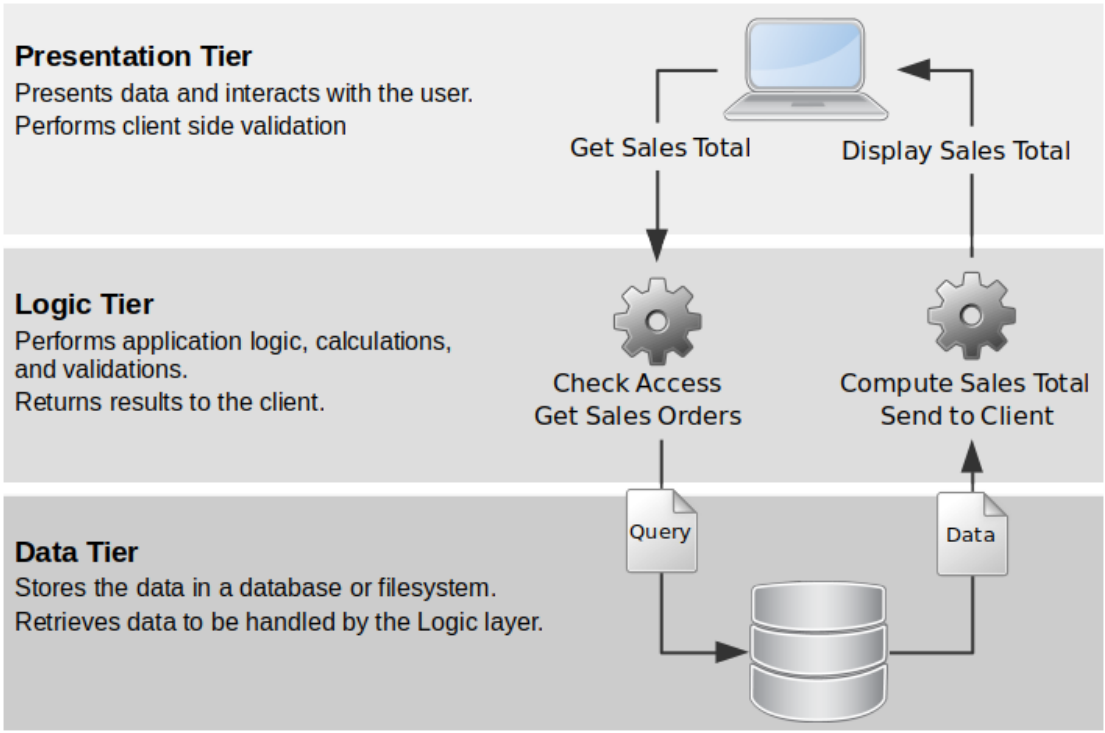

Figura 3 Arquitectura Odoo

Como hemos observado la arquitectura de Odoo sigue el patrón MVC(Modelo Vista Controlador), modelo multicapa usado por muchas aplicaciones informáticas, debido a su gran eficiencia y facilidad.

4. Análisis

Una vez ya hemos realizado el estado del arte de la tecnología que vamos a usar, es el momento de explicar el análisis, empezando con una especificación de requisitos de acuerdo con el estándar IEEE 830 y un finalizando con un ejemplo de uso de la aplicación.

Especificación requisitos software

4.1 Introducción

Aquí se va a redactar la Especificación de requisitos software (ERS) con respecto al estándar IEEE 830 [5], y como este dicta en la introducción se va a hablar del propósito, ámbito del sistema, Definiciones, Acrónimos y Abreviaturas.

Propósito

El propósito de esta ERS es la de dejar constancia del software requerido y usado para que los futuros clientes no tengan ningún tipo de problema a la hora del uso.

Ámbito del sistema

Como ya se ha mencionado anteriormente este programa se encargará de la gestión de una academia con la cual los alumnos serán capaces de inscribirse tanto a la academia como a las clases que ellos consideren oportunos, además de dar de alta a los profesores necesarios para la academia, y de crear, borrar y editar las clases.

Definiciones Acrónimos y Abreviaturas

Aquí se va a hablar de los términos utilizados en el ERS:

- **ERP**: Explicado en la sección 2.1
- **PostgreSQL**: Sistema de administración de bases de datos relacionales gratuito y de código abierto que enfatiza la extensibilidad y el cumplimiento de SQL[7].
- **XML**: Lenguaje de marcado el cual se encarga de definir reglas para poder codificar los documentos necesarios en formato legible tanto por los humanos como por las máquinas[11]
- **Python**: Python es un lenguaje de programación cuya filosofía el la de enfatizar su legibilidad del código mediante el uso necesario de la sangría. Este posee un enfoque orientado a objetos, el cual tiene como objetivo principal el de escribir un código generalmente sencillo y siempre claro y lófico para todos sus proyectos.[17]
- **Github:** Github es una forja para alojar proyectos mediante el control de versiones Git[20], será necesario para llevar los distintos proyectos al día**.**

4.2 Descripción general

El siguiente texto es común a los proyectos del equipo*: El objetivo de la aplicación es poder gestionar una academia, desde la creación de los usuarios hasta la planificación de las clases y sus actividades, voy a separar las funciones a concretar dependiendo de que miembro del equipo la ejecuta como hemos visto en el punto 2 y estas son:*

Yo	Jinye	Sergio
Dar de alta, de baja y editar alumnos. *Dar de alta, de baja y editar profesores.* *Suscribir y desuscribir a los alumnos de éstas.* *Notificar la inscripción* *Dejar constancia de los precios de las clases.*	*Crear, borrar, editar y planificar las clases con sus respectivas actividades.* *Crear, borrar, editar semestres.* *Gestionar los permisos para los distintos usuarios.* *Notificar cambios realizados en la clase.*	*Controlar la asistencia a las clases.* *Almacenar el registro de la asistencia a las clases por parte de los profesores.* *Notificación de asistencia.*

Tabla 3: Funciones equipo

Características de los usuarios

El uso de software de nuestra aplicación va a estar dirigido para los tres tipos de usuarios que puede haber. El alumno va a ser dotado con menos permisos de los tres, siendo solo capaz de acceder al calendario, ver sus clases y registrarse a la que crea conveniente, este no será capaz ni de acceder a la lista de usuarios, ni modificar ningún elemento.

El profesor a diferencia del alumno va a ser capaz de acceder a la lista de los usuarios, pudiendo tanto crear como modificar o borrar usuarios. Además de esto el profesor también será capaz de crear las clases de la academia y editarlas como guste.

Para finalizar el administrador es el usuario que más permisos tiene, teniendo los mismos permisos que el profesor, además de poder modificar los ajustes de Odoo, pudiendo así modificar los usuarios desde la base de datos de Odoo, o incluso instalar o desinstalar otros módulos.

Restricciones

La aplicación requiere de ciertas condiciones para su correcto funcionamiento, las que son:

1. Máquina (tanto física como virtual) con sistema operativo Ubuntu 20.04 o superior.
2. Navegador web actualizado (Google Chrome, Firefox, por ejemplo).
3. PostgreSQL 9.6 instalado en el sistema.
4. Python 3.9.5 con las librerías necesarias para las dependencias de Odoo instalado.
5. Correcta instalación de Odoo 13 en el sistema.

Cabe resaltar también que este proyecto ha sido realizado usando Odoo 13, por tanto, la posible actualización de este podría suponer fallos en el módulo ocasionando posibles modificaciones, así como de Ubuntu o PostgreSQL.

Requisitos futuros

Como un futuro requisito de nuestro programa se podría destacar la instalación de un módulo de Odoo que tenga relación con las finanzas, ya que el siguiente paso la aplicación es la de implementar un sistema con facturas el cual sean enviados al cliente cuando este se inscriba en una clase.

4.3 Requisitos específicos:

Aquí se va a detallar los requisitos necesarios para que se puedan realizar las pruebas y verificar su cumplimiento. (Como he dicho anteriormente la aplicación es un trabajo conjunto, por tanto, esta va a constar de más requisitos de los que van a ser descritos, puesto que solo voy a referenciar los que he realizado yo además de inicio y cierre de sesión ya que son comunes) Los requisitos específicos se van a mostrar en forma de tabla y contarán con el siguiente contenido:

Identificación: Código que se usará como referencia de cada requisito funcional.

Nombre: Denominación que se le otorga al requisito.

Descripción: Breve explicación del requisito.

Entrada: Condición que se debe dar antes de poder ejecutar el requisito.

Proceso: Acciones realizadas por el sistema durante la ejecución del requisito.

Salida: Resultado de la ejecución del requisito.

Identificación	R1
Nombre	Inicio de sesión
Descripción	Es necesario que el sistema permita iniciar sesión con un usuario para el uso de la aplicación
Entrada	Se requiere que Odoo este iniciado para el inicio de sesión, los datos que se deben introducir son correo y contraseña.
Proceso	El sistema comprueba los datos introducidos
Salida	Si los datos son correctos se permite usar la aplicación, en caso contrario pedirá reintroducir los datos

Tabla 4: Restricción 1

Identificación	R2
Nombre	Añadir un alumno
Descripción	Se añade un nuevo alumno al sistema

Entrada	Se introducen los datos necesarios para poder introducir al alumno en la base de datos.
Proceso	El sistema introduce en la base de datos al nuevo usuario como alumno
Salida	El alumno se añade a la base de datos de Odoo.

Tabla 5: Restricción 2

Identificación	R3
Nombre	Listar alumnos
Descripción	Obtenemos una lista con los alumnos de la academia
Entrada	Se debe iniciar sesión para poder listar a los alumnos.
Proceso	El sistema busca en una base de datos los usuarios especificados como alumnos de la academia y los muestra
Salida	Se ve la lista con todos los alumnos.

Tabla 6: Restricción 3

Identificación	R4
Nombre	Editar alumno
Descripción	Se edita la información previamente introducida de un alumno.
Entrada	Desde la lista de alumnos seleccionamos el alumno que se desea editar, una vez hecho esto se permitirá editar los datos introduciendo otros nuevos.
Proceso	El sistema comprueba si los nuevos datos introducidos son correctos, y si así es aplica los cambios a la base de datos, en caso de error notifica con un mensaje.
Salida	Se ve la lista de usuarios con los datos del alumno ya cambiados.

Tabla 7: Restricción 4

Identificación	R5
Nombre	Listar profesores
Descripción	Obtenemos una lista con los profesores de la academia

Entrada	Se debe iniciar sesión para poder listar a los profesores.
Proceso	El sistema busca en una base de datos los usuarios especificados como profesores de la academia y los muestra
Salida	Se ve la lista con todos los profesores.

Tabla 8: Restricción 5

Identificación	R6
Nombre	Crear profesor
Descripción	Se crea un nuevo profesor en el sistema
Entrada	Se introducen los datos necesarios para poder introducir al profesor en la base de datos.
Proceso	El sistema introduce en la base de datos al nuevo usuario como profesor.
Salida	Si los datos introducidos son correctos el profesor se añade a la base de datos.

Tabla 9: Restricción 6

Identificación	R7
Nombre	Editar profesor
Descripción	Se edita la información previamente introducida de un profesor.
Entrada	Desde la lista de alumnos seleccionamos el profesor que se desea editar, una vez hecho esto se permitirá editar los datos introduciendo otros nuevos.
Proceso	El sistema comprueba si los nuevos datos introducidos son correctos, y si es así aplica los cambios a la base de datos, en caso de error notifica con un mensaje.
Salida	Se ve la lista de usuarios con los datos del profesor ya cambiados.

Tabla 10: Restricción 7

Identificación	R8
Nombre	Notificar inscripción
Descripción	El usuario recibe un correo electrónico al inscribirse enseñando su usuario y contraseña.

Entrada	Se crea un nuevo usuario en el sistema independientemente de que sea profesor o usuario
Proceso	El sistema obtiene de los datos introducidos el correo y manda el correo con una plantilla previamente creada
Salida	En caso de error se notifica mediante un mensaje, y en caso contrario el nuevo usuario recibe un correo del sistema.

Tabla 11: Restricción 8

Identificación	R9
Nombre	Darse de alta en una clase
Descripción	Un alumno se da de alta a una clase ya creada.
Entrada	El alumno selecciona la clase que le interese y después elige cuanto tiempo quiere estar suscrita a esta.
Proceso	El sistema relaciona al alumno con las clases apareciéndole en su calendario y en el filtro de sus clases.
Salida	En caso de que no haya error el alumno estará suscrito a la clase y podrá ver a que clases esta suscrito en su calendario.

Tabla 12: Restricción 9

Identificación	R10
Nombre	Darse de baja de una clase
Descripción	El alumno se da de baja de una clase en la cual ya estaba inscrito
Entrada	Al seleccionar una clase en la cual el alumno ya este inscrito podrá darle a un botón de desuscribir para ya no estar inscrito en esta
Proceso	El sistema elimina al alumno de la clase en cuestión
Salida	El alumno deja de estar inscrito y por tanto ya no le aparecerá dicha clase en su calendario

Tabla 13: Restricción 10

Identificación	R11
Nombre	Notificar suscripción
Descripción	El alumno recibe un correo electrónico con los datos de la clase como el horario o el nombre al darse de alta en dicha clase para

Entrada	El alumno se da de alta en una clase.
Proceso	El sistema obtiene de los datos introducidos el correo y manda el correo con una plantilla previamente creada
Salida	En caso de error se notifica mediante un mensaje, y en caso contrario el nuevo usuario recibe un correo del sistema.

Tabla 14: Restricción 11

Identificación	R12
Nombre	Cerrar sesión
Descripción	Cerrar sesión de la aplicación una vez terminado su uso.
Entrada	El usuario debe haber iniciado sesión correctamente.
Proceso	El sistema cierra la sesión actual iniciada con dicho usuario.
Salida	Se muestra el menú de aplicaciones de Odoo

Tabla 15: Restricción 12

Con esto terminamos el análisis de requisitos específicos y ahora vamos a ver un ejemplo de uso concreto

4.4 Ejemplo de uso concreto

Se desea implementar Akademy en un conservatorio de música el cual cuenta con diversas clases que se imparten a lo largo del día en diferentes horarios. Este conservatorio requiere de una aplicación cuyos usuarios sean los profesores y los alumnos, además de un o dos administradores de la aplicación que se encargaran de gestionar la aplicación y sus ajustes internos. La aplicación requiere de las siguientes funcionalidades:

1-Almacenamiento de los alumnos en la aplicación: Los alumnos deben estar almacenados en la base de datos además de en la lista, los datos esenciales que se requieren de los usuarios son su DNI, nombre, correo electrónico, teléfono, fecha nacimiento y dirección. Estos datos deben ser editables o eliminables por un profesor siempre que se desee.

2-Almacenamiento de los profesores en la aplicación: Igual que los alumnos, los profesores deben estar guardados y poder ser mostrados, para guardar estos se requerirán de los mismos datos, y también podrán ser editados o eliminados si así se quisiese.

3-Creación de clases musicales: La aplicación deberá de poder crear una clase en cualquier horario de 17:00 a 21:00 como la duración de estas. Las clases podrán ser de la especialidad que se desee: piano, flauta, violín… Además, se deberá de fijar un número máximo de alumnos a los que puedan apuntarse a dicha clase, ya que el espacio es limitado.

4- Control de asistencia: La aplicación deberá tener un control de asistencia que compruebe si los alumnos asisten a las clases que están suscritos o no.

5- Notificación por correo: Se desea que la aplicación notifique por correo a los nuevos usuarios que se registren en la academia.

Además de esto he desarrollado diagramas de casos de uso para un mayor análisis y un prototipado de las vistas más importantes del módulo.

4.5 Diagramas de casos de uso

Un diagrama de casos de uso es una descripción gráfica de las posibles interacciones de un usuario con un sistema. Un diagrama de casos de uso muestra varios casos de uso y diferentes tipos de usuarios que tiene el sistema y, a menudo, también irá acompañado de otros tipos de diagramas[9]. Para realizar los siguientes diagramas de casos de uso voy a seguir el estándar UML (Unified Modeling Language), destacar que voy a usar la web de Ludichart [10] para la realización de estos

Para empezar en la figura 4.1 se puede observar el diagrama de caso de un alumno el cual entre a la aplicación, primero debe iniciar sesión y una vez hecho esto podrá o cerrarla y salir o acceder al calendario desde el cual tendrá acceso a las clases y podrá suscribirse a las que guste o desuscribirse de las que ya no le interese.

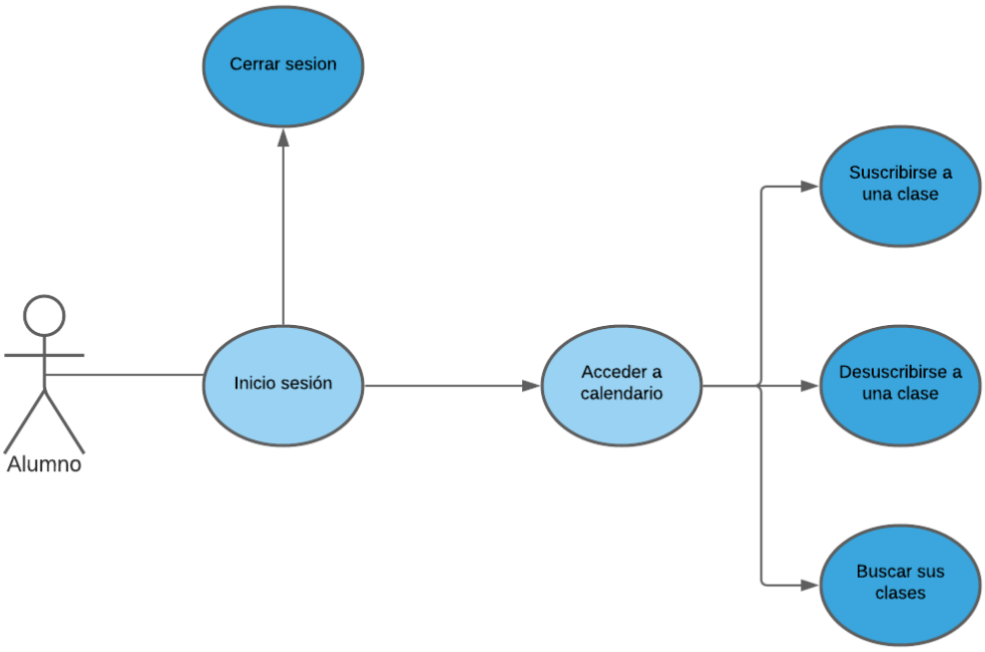

Figura 4: Diagrama de casos de uso alumno

Si por otra parte inicia sesión un profesor este podrá acceder tanto a la lista de usuarios como al calendario de clases, desde la lista de los usuarios tendrá potestad para crear a un nuevo usuario o editar/borrar un usuario ya existente, por otra parte, si accede al calendario de clases desde aquí podrá crear editar o borrar clases. Estas funciones se ven reflejadas en la figura 4.2.

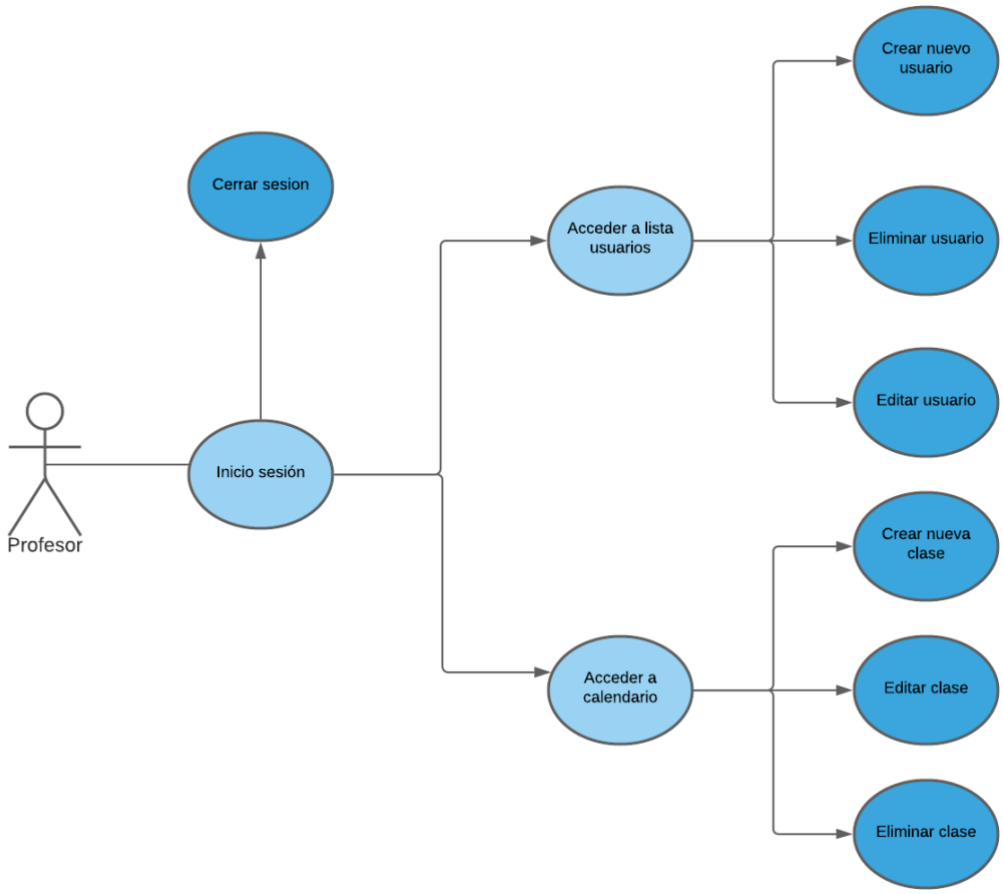

Figura 5: Diagrama de casos de uso profesor

Para finalizar con el caso de uso, desde la vista de un administrador este podrá realizar todo lo que realiza el profesor con la adición de poder modificar usuarios como otorgando o denegando permisos a otros usuarios o borrando algunos desde la base de datos de Odoo de res.users. Este diagrama se ve reflejado en la figura 4.3.

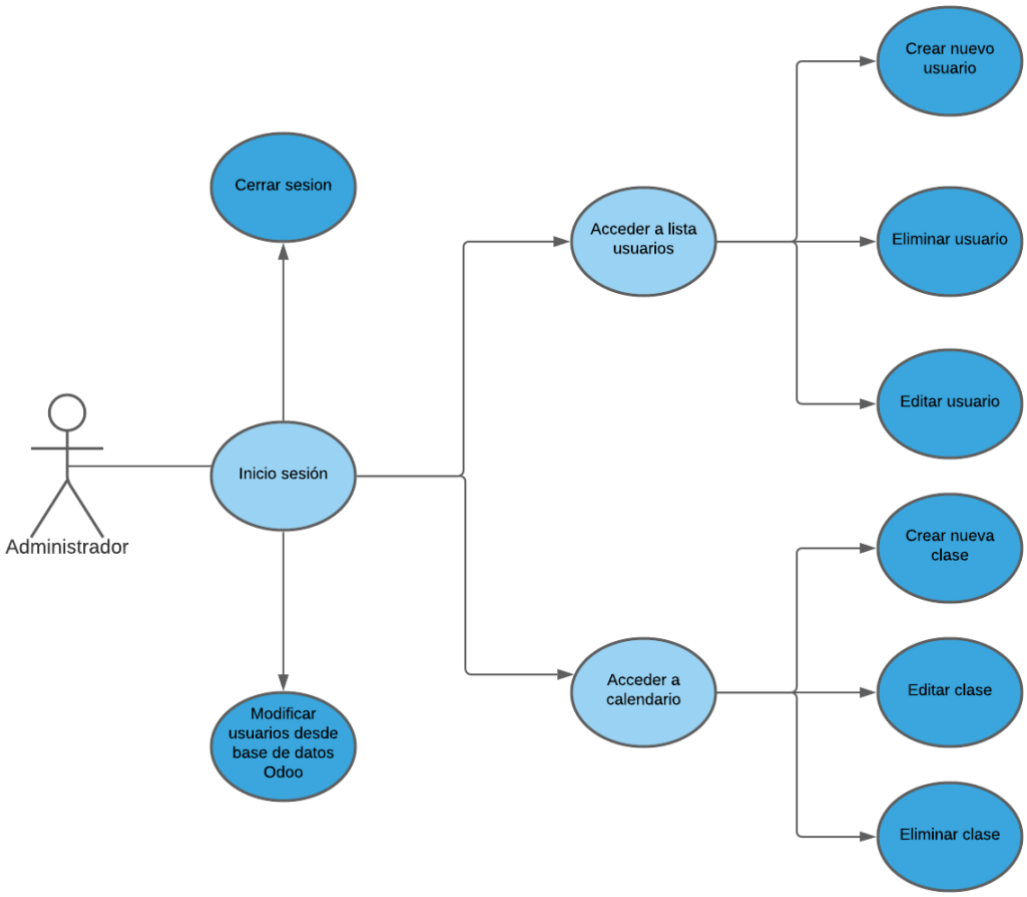

Figura 6: Diagrama de casos de uso administrador.

4.6 Prototipado

Para finalizar el diseño de la solución he realizado prototipos de la interfaz del usuario de las vistas más comunes en mi aplicación, los cuales van a ser la vista de la lista de los usuarios, la inserción de un nuevo usuario y finalmente la ventana que se abre cuadno un alumno se suscribe

a una clase.Cabe destacar también que estos prototipos estan sujetos a cambios y los tres han sido realizado a mano.

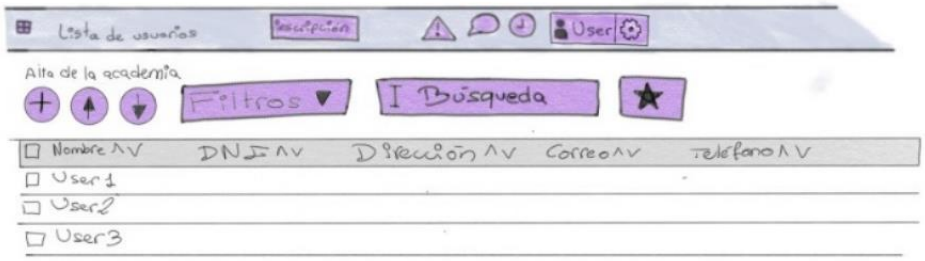

Figura 7: Prototipo lista usuarios

La vista de la lista de usuarios va a estar formada por una tabla principal en la cual se vean todos los usuarios además de sus datos personales, además de esto arriba de la tabla va a existir botones para crear usuario además de para poder buscar uno en concreto por nombre o añadir filtros.

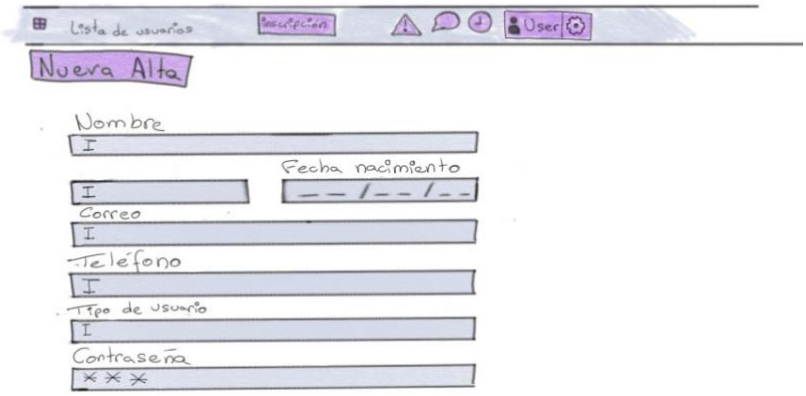

Figura 8: Prototipo introducir datos usuarios

La vista tanto para introducir a un nuevo usuario como para editar uno ya existente va a ser una muy simple en la que simplemente habrá un campo para rellenar por cada dato que sea necesario para la inscripción de un usuario.

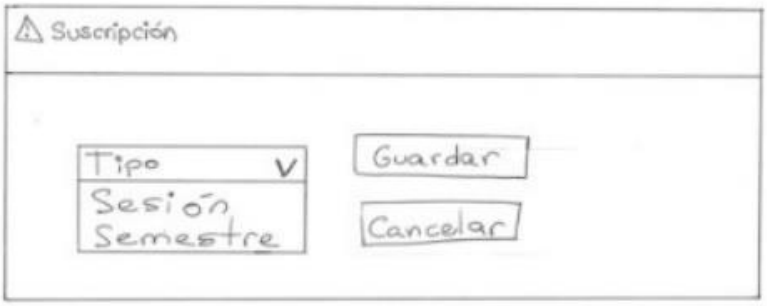

Figura 9: Prototipo suscribir alumno

Finalmente, la ventana por la cual un alumno será capaz de suscribirse a una clase va a ser muy intuitiva, el alumno solo tendrá que decidir el tiempo que va a estar suscrito a la clase, en este caso si va a ser solo esa sesión, o de todo el semestre, más adelante se planea añadir más opciones.

5. Diseño de la solución

Una vez realizado el análisis del problema es el momento de realizar el diseño de la solución, para esto se va a realizar un diagrama de clases con los objetos del proyecto.

5.1 Diagrama de clases

Un diagrama de clases es un diagrama con estructura estática el cual es capaz de describir la estructura de un sistema mediante sus clases, atributos, métodos y las relaciones que existen entre sus objetos[19]. Para la realización de este diagrama también voy a seguir el estándar UML.

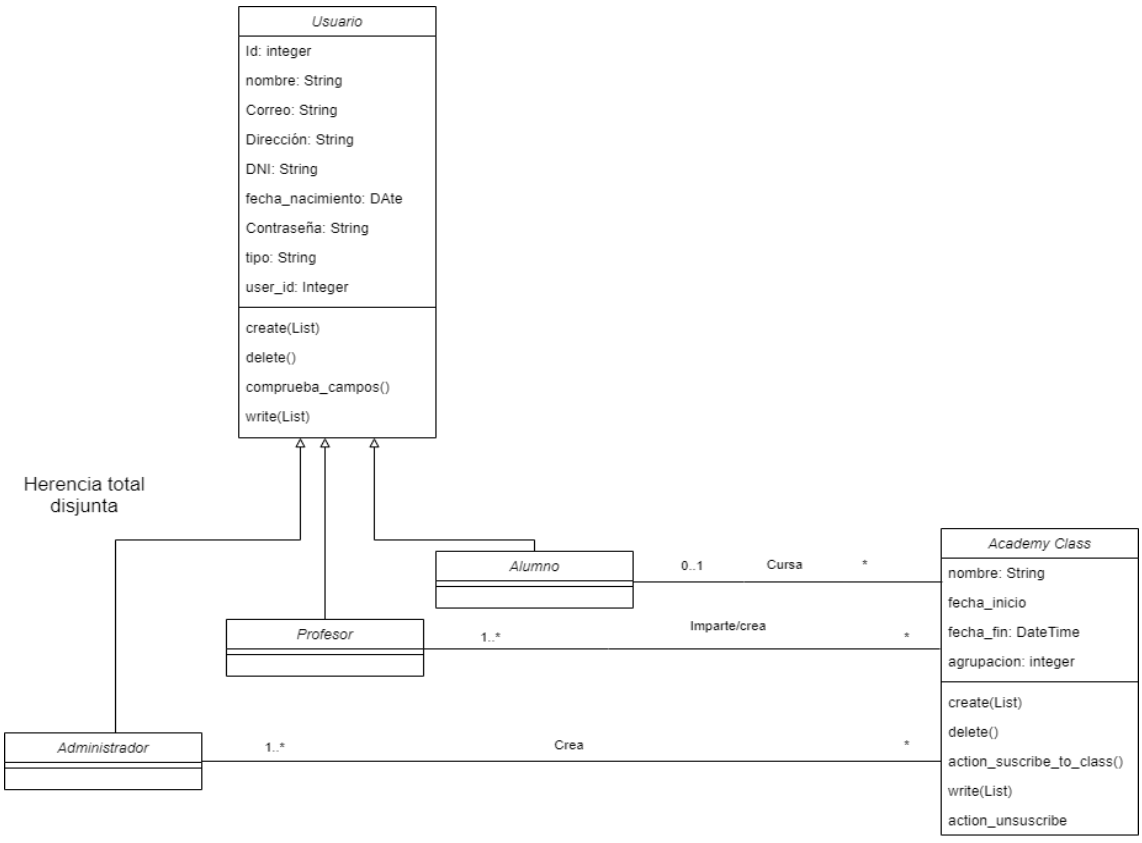

Figura 10: Diagrama de clases

6. Desarrollo de la solución

En este apartado se va a explicar el desarrollo del módulo que he desarrollado que como se ha mencionado anteriormente se centra en la inscripción de usuarios. Para comenzar en la imagen 8 se pueden ver los directorios de lo que esta formado mi módulo, los cuales se explican a continuación:

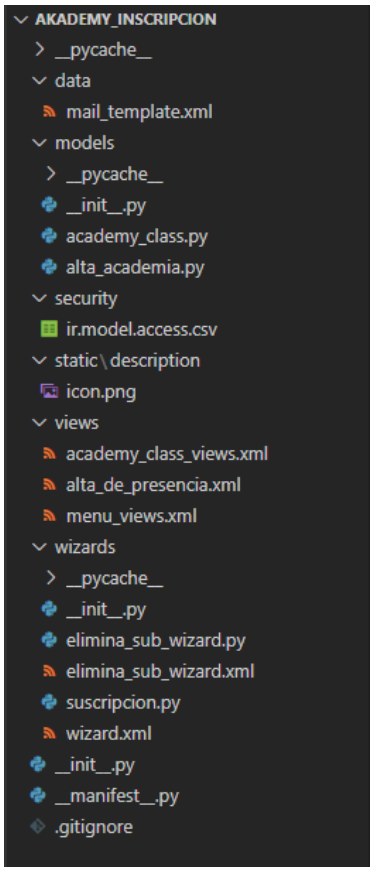

Figura 11: Estructura clases

-**Akademy_users**: Directorio el cual contiene todos los subdirectorios y archivos de los que esta formado nuestro addon.

__pycache__: directorio el cual contiene archivos de caché de código de bytes que python genera de manera automática (no es relevante para el módulo).

Data: Directorio que contiene las plantillas que van a ser usadas en el módulo, en este caso solo será usada una plantilla de correo.

models: Directorio el cual contiene los objetos de nuestro módulo.

security: Directorio el cual contiene la configuración de la seguridad del módulo.

static: Directorio el cual contiene datos estáticos, en nuestro caso solo va a contener un icono para la aplicación, aunque podría contener otros documentos como archivos CSS, datos CSV….

views: Directorio el cual contiene las vistas del módulo en formato XML.

wizards: Directorio en el que como el nombre indica contiene los wizards del módulo los cuales son modelos que muestran una vista de formulario al usuario generalmente en una ventana de diálogo[12]

Una vez explicado a rasgos generales de que se compone nuestro módulo es el momento de explicar en profundidad los archivos de este.

6.1 Modelos

Los modelos como he explicado previamente son clases de Phyton los cuales implementan los objetos de nuestra aplicación. Para este módulo he creado dos modelos: academy_class.py y alata_academia.py.

Para empezar el archivo __init__.py es un inicializador necesario de los otros modelos creados, por tanto, los importa como se ve a continuación.

__init__.py

```
from . import alta_academia
from . import academy_class
```

También se ha creado otro modelo denominado academy_class.py el cual se encarga de abrir la ventana del wizard que veremos más adelante cuando un alumno decide suscribirse a una clase.

academy_class.py

```
from odoo import api,models,fields, _

class AcademyClass(models.Model):
    _inherit = 'academy.class'

    def action_suscribe_to_class(self):

        return{
        'type': 'ir.actions.act_window',
        'name': 'Suscripción',
        'view_mode': 'form',
        'view_type':'form',
        'res_model': 'suscripcion',
        'view_id':self.env.ref('Odoo_Alejandro.crea_sub_wizard').id,
        'target':'new'
        }
```

Por último El archivo alta_academia.py, el cual contiene los atributos y métodos del objeto usuario, estos objetos son el nombre, DNI, dirección, correo, teléfono, fecha_nacimiento, contra,user_id y tipo_usuario. Todos los atributos son auto explicativos a excepción quizá de tipo, el cual indica si el usuario va a ser profesor o alumno. La creación de estos atributos se ve en el siguiente código:

alta_academia.py

```
from odoo import api,models,fields, _
from datetime import datetime, timedelta
from dateutil.relativedelta import relativedelta
from odoo.exceptions import ValidationError

class AltaAcademia(models.Model):
    _name = "alta.academia"


    nombre = fields.Char(
        string = 'Nombre'
    )
    DNI = fields.Char(
        string = 'DNI'
    )
    direccion = fields.Char(
        string = 'Dirección'
    )
    correo = fields.Char(
        string = 'Correo'
    )
    telefono = fields.Integer(
        string = 'Telefono'
    )
    fecha_nacimiento = fields.Date(
        string = 'Fecha nacimiento'
    )
    contra = fields.Char(
        string = 'Contraseña'
    )
    user_id=fields.Integer(
        string='ID'
    )
    tipo_usuario = fields.Selection(
    [
        ('profesor', 'Profesor'),
        ('estudiante', 'Estdiante')
    ],
        string="Tipo"
    )
```

Además, hay que tener en cuenta que a la hora de crear usuario existen restricciones pertinentes, por ejemplo, el DNI o el número de teléfono deben ser posibles, en caso de que estén mal introducidos saltará un error y no dejará crearlos como se puede ver en este código:

```
def comprueba_campos(self):
      for rec in self:
          if len(str(rec.telefono)) != 9:
              raise ValidationError(_('Número de telefono no válido'))
          elif len(rec.DNI)!=9:
              raise ValidationError(_('Número de DNI no válido'))
```

6.2 wizards

El directorio wizards está formado por un __init__py, la cual, al igual que en modelos se encarga de importar los archivos Python, en este caso suscripción.py y elimina_sub_wizard.py, archivos los cuales también están dentro del directorio wizard, los cuales se encargan de que sea posible para un alumno suscribirse en el caso de wizard.py y desuscribirse para elimina_sub_wizard.py.

__init__.py

```
from . import suscripcion
from . import elimina_sub_wizard
```

suscripción.py

```
from odoo import models, fields, api, _

class Suscripcion(models.TransientModel):
    _name='suscripcion'
    #inherit='alta.academia'
    type = fields.Selection(
    [
        ('sesion', 'Sesión'),
        ('semester', 'Semestre')
    ],
        string="Tipo"
    )


    def action_button_confirm(self):
        academy_id = self.env.context['active_id']
        academy = self.env['academy.class'].browse(academy_id)
        if self.type == 'sesion':
```

```
            academy.sudo().suscribed_people = [(4, self.env.uid, False)]
        else:
            if academy.semester_id:
                same_semester_classes = self.env['academy.class'].search([('name',
               '=', academy.name),('semester_id', '=', academy.semester_id.id)]
                for subject in same_semester_classes:
                    subject.sudo().suscribed_people = [(4, self.env.uid, False)]
            else:
                academy.sudo().suscribed_people = [(4, self.env.uid, False)]
```

elimina_sub_wizard.py

```
from odoo import models, fields, api, _

class elimina_sub_wizard(models.TransientModel):
    _name='desuscripcion'
    #inherit='alta.academia'
    type = fields.Selection(
    [
        ('sesion', 'Sesión'),
        ('semester', 'Semestre')
    ],
        string="Tipo"
    )


    def action_button_confirm(self):
        academy_id = self.env.context['active_id']
        academy = self.env['academy.class'].browse(academy_id)
        if self.type == 'sesion':


            academy.sudo().suscribed_people = [(3, self.env.uid, False)]
        else:
            if academy.semester_id:
                same_semester_classes = self.env['academy.class'].search([('name',
                '=', academy.name),('semester_id', '=', academy.semester_id.id)])
                for subject in same_semester_classes:
                    subject.sudo().suscribed_people = [(3, self.env.uid, False)]
            else:
                academy.sudo().suscribed_people = [(3, self.env.uid, False)]
```

También se encuentra en este directorio el archivo wizard.xml y elimina_sub_wizard.xml que como su nombre indica son las vistas del wizard, pero estos se van a ver en el apartado de vistas.

6.3 Vistas

Aquí se encuentran todas las vistas de nuestro módulo en XML. Los principales tipos de vistas que ofrece Odoo son *form*, *list* y *tree*, aunque también existen otros como calendar, *graph* o *kanban*[18]. Yo he realizado vistas de tipo *form*, *tree*.

Mi view principal es alta_de_presnecia.xml la cual se compone de un formulario *form* y de un árbol *tree*, inscripción_form es la vista que se muestra cuando se van a introducir los datos de un nuevo usuario estos son los mismos que se especifican alta_academia.py y como se puede ver con el "required = 1" todos estos campos son deben estar rellenados para la creación del usuario. Por otra parte, inscripción_tree muestra la lista de usuarios ya creados, en este caso se muestran los mismos datos para inscribir, excepto el campo de la contraseña por razones obvias.

alta_de_presnecia.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<odoo>
 <record id="inscripcion_form" model="ir.ui.view">
    <field name = "name">alta.academia.form</field>
    <field name ="model">alta.academia</field>
    <field name="arch" type="xml">
      <form string="inscripcion">
        <group id="id_suscripcion_form">
          <field name="nombre" required="1"/>
          <field name="DNI" required = "1"/>
          <field name="direccion" required = "1"/>
          <field name="correo" required = "1"/>
          <field name="contra" required = "1" password="True"/>
          <field name="telefono" required = "1"/>
          <field name="fecha_nacimiento" required = "1"/>
          <field name ="tipo_usuario" required="1"/>
        </group>
      </form>
    </field>
  </record>
  <record id="inscripcion_tree" model="ir.ui.view">
    <field name = "name">alta.academia.tree</field>
    <field name ="model">alta.academia</field>
    <field name="arch" type="xml">
      <tree>
          <field name="nombre" required="1"/>
          <field name="DNI" required = "1"/>
          <field name="direccion" required = "1"/>
          <field name="correo" required = "1"/>
```

```
          <field name="telefono" required = "1"/>
          <field name="fecha_nacimiento" required = "1"/>
          <field name ="tipo_usuario" required="1"/>
      </tree>
    </field>
  </record>
</odoo>
```

Mi segunda vista más importante es menu_views.xml, la cual muestra la pantalla principal de mi módulo siendo el título “Lista usuarios”.

menu_views.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<odoo>
  <record id="accion_menu_item" model="ir.actions.act_window">
    <field name="name">Alta de la academia</field>
    <field name="res_model">alta.academia</field>
    <field name="view_mode">tree,form</field>
  </record>
  <menuitem id="raiz_academia" name="Lista usuarios"/>
  <menuitem id="academia" parent="raiz_academia" action="accion_menu_item"
   name="in scripcion"/>
</odoo>
```

También existen otras vistas suplementarias, siendo academy_class_views.xml, wizard.xml, elimina_sub_wizard.xml y mail_template.xml. La primera de estas se encarga de heredar la vista de la información de la clase realizada por mi compañero e insertar en esta un botón de suscribirse para que los alumnos puedan suscribirse a las clases que quieran, wizard.xml como he mencionado anteriormente es la vista del wizard que se abre una vez pulsas en el botón de suscribirse siendo esta una vista muy sencilla con solo el campo de tipo el cual se elige cuanto tiempo vas a estar el alumno suscrito a una clase, pero más adelante se le añadirán otras funcionalidades como precios u horarios concretos, elimina_sub_wizard.xml es igual que wizard.xml solo que se abre cuando pulsas el botón de desuscribirse, botón que solo es visible cuandlo el alumno ya está suscrito a la clase. Por último mail_template.xml muestra la vista de la plantilla que tiene el correo que se manda a los alumnos cuando estos se registran en la aplicación.

El código de estas tres vistas se muestra a continuación:

academy_class_views.xml

```
<?xml version="1.0" encoding="UTF-8"?>

<odoo>
```

```
    <record id="academy_class_form_view" model="ir.ui.view">
        <field name="name">academy.class.form</field>
        <field name="model">academy.class</field>
        <field name="inherit_id" ref="odoo_jinye.academy_class_form_view"/>
        <field name="arch" type="xml">
            <xpath expr="//sheet" position="before">
              <header>
                <button name="action_suscribe_to_class" string="Suscribirse" type="
                object"/>
              </header>
            </xpath>
        </field>
    </record>
</odoo>
```

wizard.xml

```
<odoo>
  <record id="crea_sub_wizard" model="ir.ui.view">
    <field name="name">crear.suscripcion.wizard</field>
    <field name="model">suscripcion</field>
    <field name="arch" type="xml">
      <form string="Confirmar subscripcion">
        <group>
          <field name ="type"/>
        </group>
        <footer>
          <button name="action_button_confirm" string="Confirmar" type="object"
           class="bin-primary"/>
          <button string="Cancelar" class="btn-secondary" special = "cancel"/>
        </footer>
      </form>
    </field>
  </record>
  <record id="accion_abre_ventana" model="ir.actions.act_window">
    <field name="name">Crea Suscripcion</field>
    <field name="type">r.actions.act_window</field>
    <field name="res_model">suscripcion</field>
    <field name = "view_mode">form</field>
    <field name="view_id" ref="crea_sub_wizard"/>
    <field name="target">new</field>
  </record>
</odoo>
```

Elimina_sub_wizard.xml

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<odoo>
  <record id="elimina_sub_wizard" model="ir.ui.view">
    <field name="name">elimina.suscripcion.wizard</field>
    <field name="model">desuscripcion</field>
    <field name="arch" type="xml">
      <form string="Confirmar desubscripcion">
        <group>
          <field name ="type"/>
        </group>
        <footer>
          <button name="action_button_confirm" string="Confirmar" type="object"
class="oe_highlight" />
          <button string="Cancelar" class="btn-secondary" special = "cancel"/>

        </footer>
      </form>
    </field>
  </record>
  <record id="accion_abre_ventana_desus" model="ir.actions.act_window">
    <field name="name">Crea Desuscripcion</field>
    <field name="type">r.actions.act_window</field>
    <field name="res_model">desuscripcion</field>
    <field name = "view_mode">form</field>
    <field name="view_id" ref="crea_sub_wizard"/>
    <field name="target">new</field>
  </record>

</odoo>
```

mail_template.xml

```
<?xml version ="1.0" ?>
<odoo>

    <record id = "correo" model = "mail.template">

      <field name = "name"> Enviar confirmación</field>
      <field name = "model_id" ref="Odoo_Alejandro.model_alta_academia"/>
      <field name = "email_from"> ${object.nombre_email_formatted |safe}</field>
      <field name = "email_to">${object.correo.id}</field>
      <field name = "subject">alumno carta${object.DNI}</field>
      <field name = "body_html" type="html">
        <div style = "margin: 0px; padding; 0px;">
          <p style = "margin: 0px; padding; 0px; font-size: 13px;">
            Buenos días, ${object.nombre}
            <br /><br />
```

```
    Se ha realizado su inscripcion correctamente su contraseña es ${object.contra}
            <br /><br />
            Saludos
          </p>
        </div>
      </field>
      <!-- <field name="report template" ref="model_account_invoices"/> -->
      <field name ="report_name">Estudiante${(object.nombre)}</field>
    </record>
</odoo>
```

6.4 Seguridad

También cabe resaltar el archivo ir.model.acces.csv en la carpeta de security, el cual se encarga de adjudicar los permisos a los distintos tipos de usuarios que hay en la academia, otorgando total permiso para profesores y administradores, y ningún permiso de mi módulo a los alumnos, ya que estos no deben ver nunca la lista de usuarios[14].

ir.model.acces.csv

```
id,name,model_id:id,group_id:id,perm_read,perm_write,perm_create,perm_unlink
access_TFG_admin,TFG.task.user,model_alta_academia,odoo_jinye.group_academy_administrartor,1,1,1,1
access_TFG_student,access_academy_TFG.student,model_alta_academia,odoo_jinye.group_academy_student,0,0,0,0
access_TFG_teacher,TFG.task.user.teacher,model_alta_academia,odoo_jinye.group_academy_teacher,1,1,1,1
```

6.5 Otros ficheros

Para finalizar existen otros dos ficheros que no se encuentran en ninguna carpeta los cuales son __init__.py y __manifest__.py, el primero como los dos anteriores solo importa otros ficheros, en este caso models y wizards, y el segundo se encarga de dar acceso a odoo a las diferentes vistas que se ejecutan, siendo imposible realizar nuestra aplicación sin cualquiera de estos dos ficheros[13].

__init__.py

```
from . import models
from . import wizards
```

__manifest__.py

```
{
    'name': 'TFG',
    'description': 'descripcion aqui',
```

```
    'author': 'Alejandro López Galiana',
    'depends': ['base','mail','odoo_jinye'],
    'data':[
        'security/ir.model.access.csv',
        'views/academy_class_views.xml',
        'views/alta_de_presencia.xml',
        'views/menu_views.xml',
        'data/mail_template.xml',
        'wizards/wizard.xml'
    ],
    'application': True,
}
```

7. Pruebas funcionales.

Una vez realizado nuestra implantación es el momento de comprobar que todo funcione como se desea, para esto se van a realizar unos casos de prueba. Estos casos de prueba van a intentar que se cumplen los requisitos nombrados en el punto 3.1.3.

Para empezar, se comprueba el correcto inicio de sesión, como se puede ven en la figura 12, voy a iniciar la aplicación con un usuario admin, el cual como su nombre indica tiene los permisos de administrador.

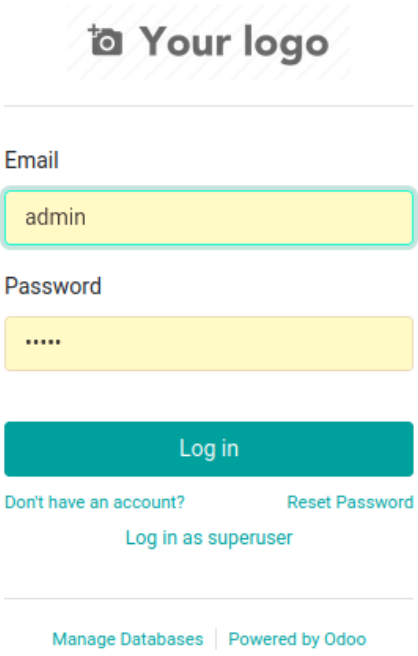

Figura 12: inicio sesión Odoo

Una vez se ha instalado nuestro modulo en Odoo ya podemos acceder a nuestra funcionalidad, para abrir la aplicación se deberá pulsar en la esquina superior izquierda y se abrirá un menú con todos los módulos instalados, figura 13, aquí abrimos el de lista usuarios, que nos mostrará cómo se puede ver en la figura 14. Una vez aquí creamos un nuevo usuario mediante el botón de crear, esto abrirá una nueva ventana la cual se ve reflejada en la figura 15, la que nos pedirá que introduzcamos los campos necesarios para crear al usuario en cuestión, en este caso vamos a crear un nuevo usuario de tipo profesor.

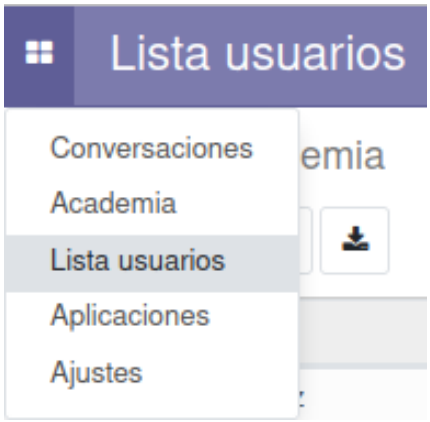

Figura 13: Menu Odoo

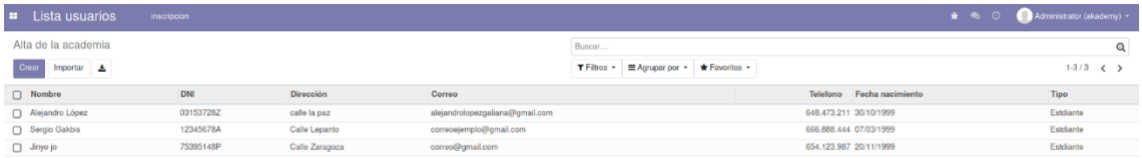

Figura 14: Lista usuarios

Figura 15: Inscripción usuario

Una vez creado iniciamos sesión con esta cuenta, y este crea una clase en la cual los alumnos podrán suscribirse, (esto no se refleja en las pruebas funcionales puesto que ha sido realizado por mi compañero) cerramos sesión con el profesor ya que no lo vamos a necesitar más e iniciamos con un alumno, en este caso Alejandro López. Una vez iniciado sesión lo primero que se puede ver es que, a diferencia de administradores y profesores, el alumno no tiene acceso a la lista de usuarios, y para suscribirse a una clase deberá pinchar en academia. Aquí se abre un calendario con sus clases automáticamente, pero como todavía no está suscrito en ninguna sale vacío, figura 16, Por tanto, quitamos el filtro que aparece en la barra de búsqueda de la esquina

superior derecha y podrá ver todas las clases ofertadas en ese momento, aquí el alumno deberá elegir en que clase desea suscribirse pulsando en ella, y después en el botón de editar figura 17

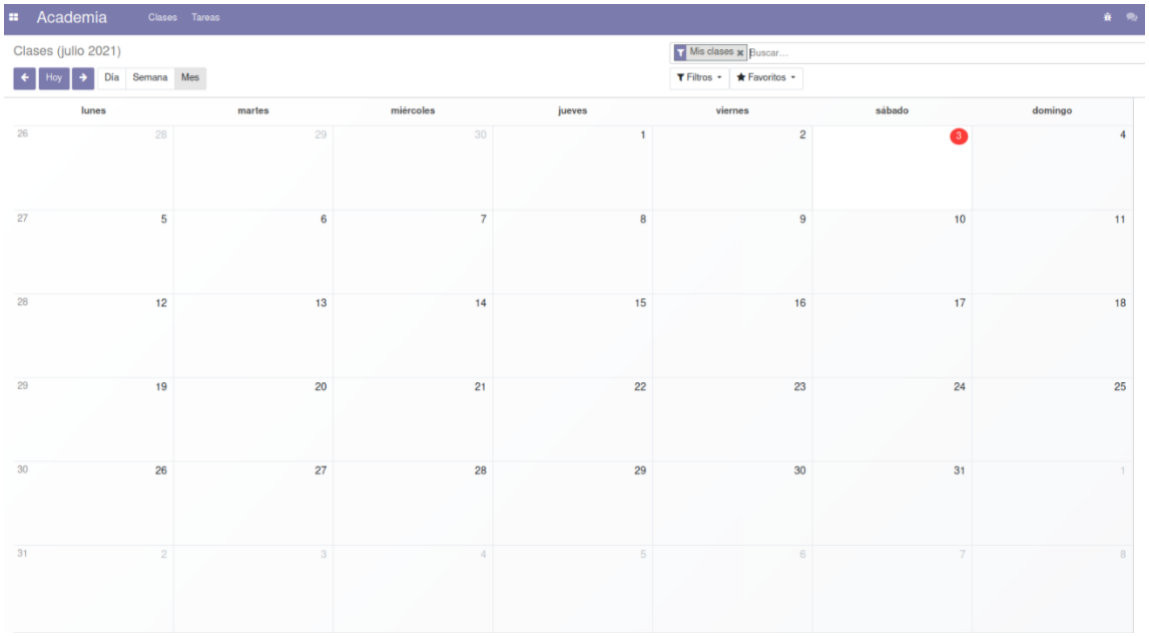

Figura 16: Calendario clases filtro

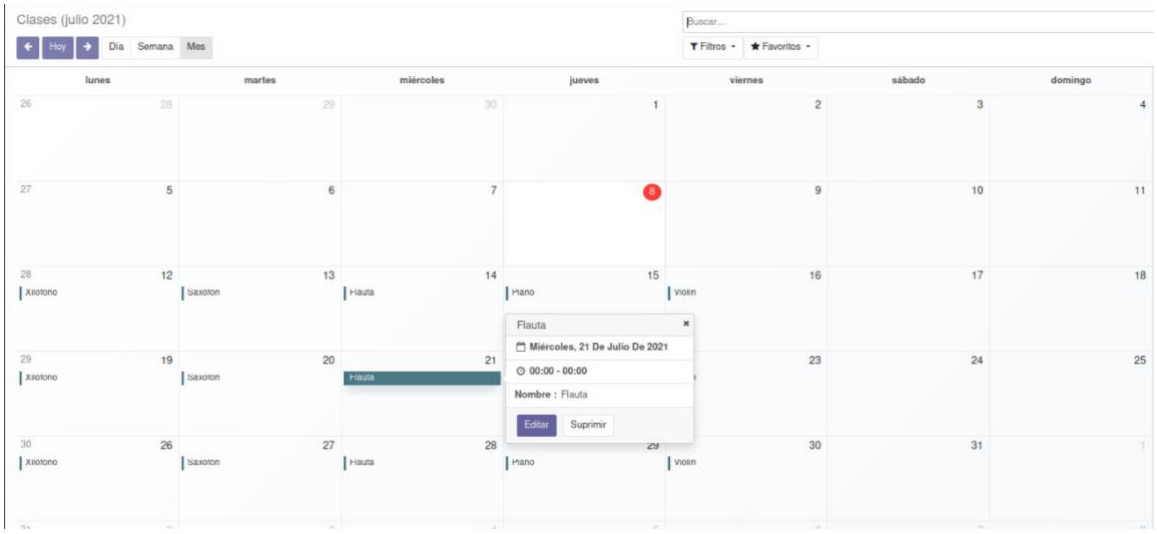

Figura 17: Calendario clases no filtro

Al abrir la clase se verá toda la información de esta introducida previamente por un profesor o administrador y en la esquina superior izquierda un botón que pone suscribirse, esto se ve reflejado en la figura 18

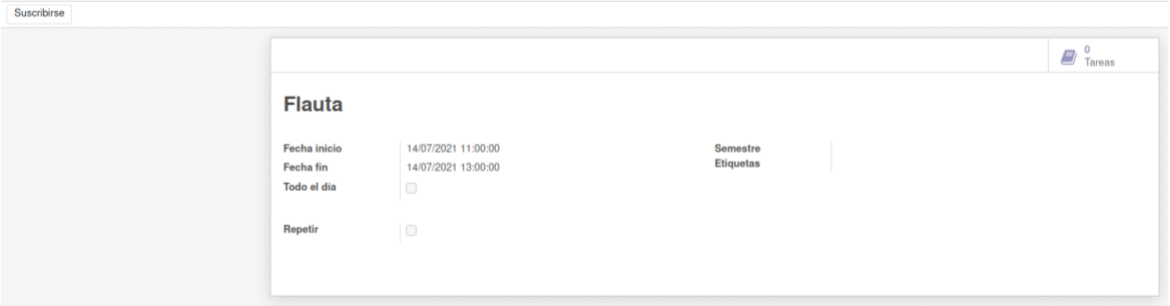

Figura 18: Información clase

Este abre una nueva ventana, en la que como se ve en la figura 19 Pedirá al usuario si desea suscribirse a la clase solamente una sesión o de forma semestral, una vez introducido esto al pulsar al botón de aceptar el alumno ya estará suscrito a la clase, esto se refleja en la figura en la lista de suscritos a la clase ya que su nombre se ha añadido.

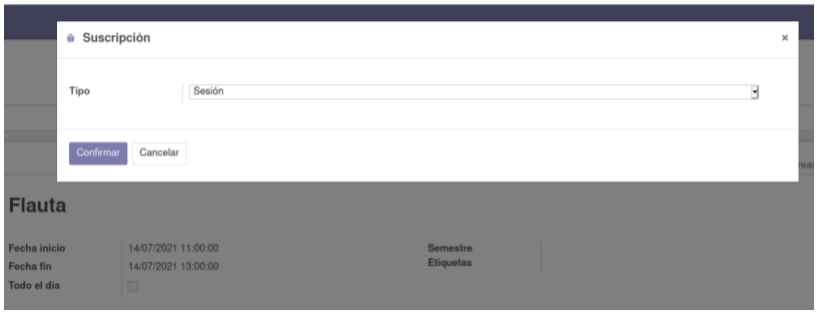

Figura 19: Suscripción clase

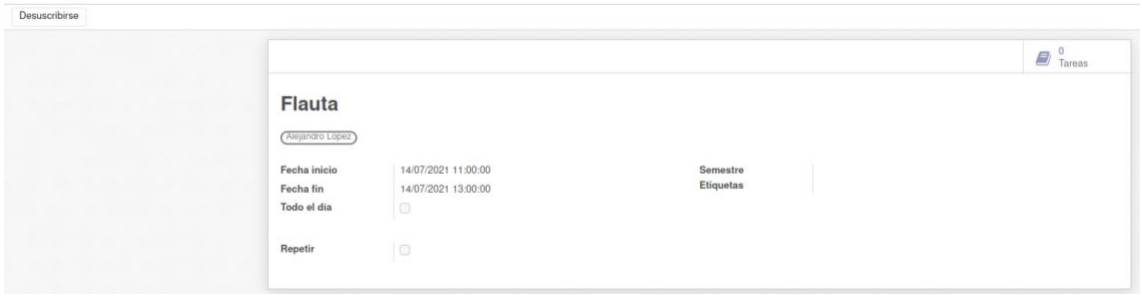

Figura 20: Información clase alumno suscrito

Una vez hecho esto si un alumno desea desuscribirse de una clase deberá realizar el mismo proceso que para suscribirse, solo que, al ya estar suscrito, es botón de suscribirse ahora es desuscribirse, figura 20. Al pulsar en este se abrirá una ventana idéntica a la de suscribirse, otra vez debiendo elegir si desuscribirse de todas las clases semestrales o de una sesión. De misma forma que suscribiéndose vemos que se ha realizado correctamente la desuscripción porque el nombre del alumno ya no aparece en la lista de usuarios suscritos, como se ve en la figura 21.

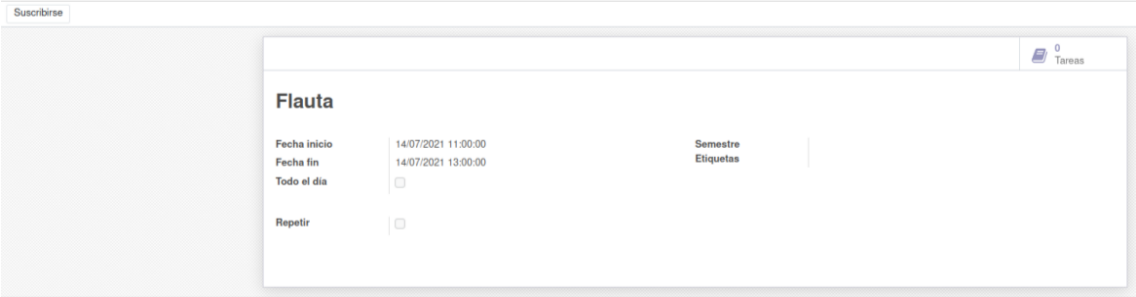

Figura 21: información clase

Con esto vemos como todas pruebas funcionales han sido realizadas con éxito, por lo que nuestro modulo ya está operativo y a la espera de su ejecución.

8. Conclusiones y proyectos futuros.

Para finalizar el proyecto se realiza una conclusión para poder valorar el proyecto ya terminado, comprobar si se han cumplido los objetivos planteados anteriormente y analizar el proyecto con todos los problemas y soluciones que se han dado durante la realización de este.

Los objetivos propuestos en la introducción han sido cumplidos satisfactoriamente, puesto que después de realizar este TFG, he aprendido mucho sobre los ERP, sabiendo ahora explicar lo que son, los principales ERP que existen, sus características, su uso en las empresas… Además de esto el objetivo principal que era aprender Odoo, ha sido completamente logrado, puesto que la aplicación es completamente funcional entendiendo el funcionamiento y las características propias de Odoo. También hay que destacar, que mediante el transcurso del proyecto he aprendido XML, lenguaje que no ha sido visto en la carrera y he mejorado mi nivel en Python.

En cuanto a los problemas que has surgido durante la realización del proyecto, podemos remarcar los siguientes:

Aprender a usar Odoo: El principal problema que me ha surgido durante la realización del TFG ha sido el de comprender como funciona Odoo y saber como utilizar sus características, esto ha sido solucionado en gran parte mediante la ayuda de páginas en internet y mención especial a un canal de YouTube denominado Odoo mates el cual me ha ayudado mucho con sus vídeos explicativos de Odoo.

Uso de XML: Nunca había realizado ningún proyecto en XML, por lo que no sabía usarlo en el momento de empezar el proyecto, pero gracias a otros ejemplos de otros módulos ya creados, pude ver de forma general como funciona el lenguaje XML, y descubrir que particularmente su uso en Odoo es muy sencillo de implementar y de programar.

TFG conjunto: El hecho de que el TFG haya sido realizado simultáneamente por tres personas, y dentro de poco otra seguirá mejorando la aplicación han surgido problemas a la hora de juntar los módulos, y que los tres funcionen completamente, la realización de esto se ha hecho mediante GitHub, creando cada uno un repositorio para nuestra parte y mediante git pull y git push actualizábamos nuestro repositorio y examinábamos lo que hacía el resto para que no surgiera ningún problema a la hora de ejecutar, realizar el trabajo de esta forma también ha implicado un ligero aumento en la dificultad a la hora de programar el módulo, porque en alguna ocasión ha hecho falta heredar de clases ajenas y la actualización de estas podía resultar en error.

Como ya he dicho el proyecto cumple con las expectativas y es completamente funcional, pero todavía quedan cosas que mejorar o implementar en la aplicación, las cuales se van a detallar a continuación:

Implementación de un sistema de pagos: Esta parte es la mejora más inmediata y necesaria de la aplicación, esta mejora, va a ser implementada por mi compañero Diego, dentro de poco, pero va a ser necesaria parte de mi módulo para que el suyo pueda funcionar completamente.

Mejora estética general: La aplicación, aunque totalmente funcional, podría ser mucho más estética en varios apartados, por lo que debería mejorarse ese apartado para una mayor vistosidad.

Actualización de Odoo: Actualmente ya existe Odoo 14.0, pero esta todavía no es usada por casi ninguna empresa, debido a que todavía esta empezando y existen errores, por lo que de momento es mejor permanecer en Odoo 13.0, pero como en el futuro será una buena mejora cambiar a esta nueva versión.

Aplicación para teléfonos móviles. Ahora mismo es posible crear aplicaciones para teléfonos, esto sería una gran mejora para la aplicación debido a que se ampliaría mucho su mercado y muchas otras empresas podrían querer utilizarla.

9. BIBLIOGRAFIA

[1] ¿Cuál es el origen de los ERP? De invento militar a software imprescindible para las empresas | Dataprix TI. *Inicio | Dataprix TI* [en línea]. [sin fecha] [consultado el 20 de abril de 2021]. Disponible en: https://www.dataprix.com/es/articulo/erp/cual-origen-erp-invento-militar-software-imprescindible-empresas

[2] COLABORADORES DE LOS PROYECTOS WIKIMEDIA. Sistema de planificación de recursos empresariales - Wikipedia, la enciclopedia libre. *Wikipedia, la enciclopedia libre* [en línea]. 23 de marzo de 2006 [consultado el 23 de abril de 2021]. Disponible en: https://es.wikipedia.org/wiki/Sistema_de_planificación_de_recursos_empresariales

[3] PANORAMA CONSULTING GROUP. 2021 ERP REPORT. Panorama consulting [en línea]. [sin fecha] [consultado el 23 de abril]. Disponible en: https://f.hubspotusercontent40.net/hubfs/4439340/Reports/ERP%20Report/2021-ERP-Report-Panorama-Consulting-Group.pdf

[4] The Odoo story. Odoo S.A. [en línea]. [sin fecha] [consultado el 27 de abril de 2021]. Disponible en: https://www.odoo.com/es_ES/blog/nuestro-blog-5/the-odoo-story-56

[5] *Especificación de Requisitos seg´un el est´andar de IEEE 830* [en línea]. 22 de octubre de 2008 [consultado el 25 de abril de 2021]. Disponible en: https://www.fdi.ucm.es/profesor/gmendez/docs/is0809/ieee830.pdf [6] Daniel Reis. Odoo 12 Development Essentials - Fourth Edition. Packt Publishing Ltd,Birmingham, cuarta edición, 2018.

[7] PostgreSQL. *PostgreSQL* [en línea]. [sin fecha] [consultado el 5 de mayo de 2021]. Disponible en: https://www.postgresql.org/

[8] Home | Odoo. *Odoo S.A.* [en línea]. [sin fecha] [consultado el 6 de mayo de 2021]. Disponible en: https://www.odoo.com/es_ES/

[9] El diagrama de casos de uso en UML. *IONOS Digitalguide* [en línea]. [sin fecha] [consultado el 8 de julio de mayo]. Disponible en: https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/diagrama-de-casos-de-uso/

[10] Software de diagrama en línea y solución visual | Lucidchart. *Lucidchart* [en línea]. [sin fecha] [consultado el 8 de mayo de 2021]. Disponible en: https://www.lucidchart.com/pages/es

[11] Extensible Markup Language (XML). *World Wide Web Consortium (W3C)* [en línea]. [sin fecha] [consultado el 10 de mayo 2021]. Disponible en: https://www.w3.org/XML/

[12]SOLÍS, Manuel Calero. Odoo de wizards y trasients. *Medium* [en línea]. 21 de octubre de 2019 [consultado el 15 de mayo de 2021]. Disponible en: https://manuelcalerosolis.medium.com/odoo-de-wizards-y-trasients-15bded28ae4f

[13] Qué es Odoo y cómo funciona. Trey, Kilobytes de Soluciones S.L. [en línea]. [sin fecha] [consultado el 15 de mayo de 2021]. Disponible en: https://trey.es/que-es-odoo-y-como-funciona

[14] COLABORADORES DE LOS PROYECTOS WIKIMEDIA. Microsoft Dynamics - Wikipedia, la enciclopedia libre. *Wikipedia, la enciclopedia libre* [en línea]. 22 de julio de 2009 [consultado el 17 de mayo de 2021]. Disponible en: https://es.wikipedia.org/wiki/Microsoft_Dynamics

[15] Infor LN: Módulos y características. ERP-Spain.com - [en línea]. [sin fecha] [consultado el 18 de mayo de 2021]. Disponible en: https://www.erp-spain.com/articulo/76108/infor/todos/infor-ln-modulos-y-caracteristicas

[16] Tipos de ERP: genéricos, parametrizados y a medida - Evaluando ERP. *Evaluando ERP* [en línea]. [sin fecha] [consultado el 1 de junio de 2021]. Disponible en: https://www.evaluandoerp.com/software-erp/conceptos-erp/tipos-de-erp/

[17] Welcome to Python.org. *Python.org* [en línea]. [sin fecha] [consultado el 7 de junio de 2021]. Disponible en: https://www.python.org/

[18] Capítulo 6 - Vistas — documentación de odoo - sphinx. *Index — documentación de odoo - sphinx* [en línea]. [sin fecha] [consultado el 7 de junio de 2021]. Disponible en: https://re-odoo-10.readthedocs.io/capitulos/vistas-disenar-la-interfaz/

[19] Tutorial de diagrama de clases UML. *Lucidchart* [en línea]. [sin fecha] [consultado el 11de junio de 2021]. Disponible en: https://www.lucidchart.com/pages/es/tutorial-de-diagrama-de-clases-uml

[20] GitHub: Where the world builds software. GitHub [en línea]. [sin fecha] [consultado el 15 de junio de 2021]. Disponible en: https://github.com/

Apéndice 1: Configuración del sistema

En este último apartado de la memoria se va a explicar la configuración necesaria para la realización del proyecto. Explicando primero como configurar una máquina virtual con el sistema operativo Ubuntu, después la instalación de Odoo 13 y para finalizar otras herramientas que han sido necesarias para la realización del proyecto.

A.1 Máquina Virtual

Para realizar la máquina virtual primero se necesita instalar una aplicación que permita ejecutar máquinas virtuales, en mi caso voy a utilizar Virtual Box, esta puede ser descargado de manera gratuita en el siguiente enlace: https://www.virtualbox.org/wiki/Downloads. Una vez hecho esto será necesario descargar la imagen ISO de la partición Ubuntu que requiramos en nuestro caso la 20.04, estas se encuentran en su web oficial: https://ubuntu.com/download/desktop.

Una vez tenemos tanto la máquina virtual como el archivo ISO, hay que configurar la máquina virtual para que pueda ejecutar el sistema operativo, para esto se le hemos dado 3072 Mb de RAM y 20 Gb de HDD. Para acabar ejecutamos la máquina virtual y la configuramos siguiendo la guía mostrada en la siguiente web: https://www.linuxtechi.com/ubuntu-20-04-lts-server-installation-guide/.

A.2 Instalación de Odoo 13.0

El primer paso para la instalación del servidor Odoo será la actualización del sistema operativo. Para ello, debe disponer de una conexión a Internet. Compruebe que su máquina virtual disponga de acceso a Internet (por ejemplo, se puede lanzar “pings” contra el servidor 8.8.8.8 y contra www.google.com.

Una vez se haya asegurado de que disponga de acceso a Internet, se debe hacer lo siguiente:

```
#sudo apt-get update
#sudo apt-get upgrade
```

Para gestionar con mayor comodidad nuestro servidor Ubuntu desde nuestro propio PC, se instala el servicio ssh (acceso remoto a consola):

```
#sudo apt-get install openssh-server
```

Antes de instalar el propio servidor Odoo, instalaremos la base de datos postgresql, ejecutando lo siguiente:

```
#sudo apt-get install postgresql
```

Ha llegado el momento de la instalación del servidor Odoo 13. Para ello haga lo siguiente:

```
#sudo wget http://nightly.odoo.com/odoo.key
# cat odoo.key | sudo apt-key add -
# sudo nano /etc/apt/sources.list
```

con este comando editaremos el fichero sources.list; al final de dicho fichero debe añadir: " deb http://nightly.odoo.com/13.0/nightly/deb/ ./ ". Guarde el fichero (CTRL+o) y salga (CTRL+x).

Posteriormente, se ejecuta lo siguiente:

```
#sudo apt-get update
#apt-cache search odoo
#apt-cache show odoo
#sudo apt-get install odoo
```

Tras unos minutos, usted tendrá instalado el servidor Odoo v13.

Para acceder al servidor, en el equipo donde ejecuta la máquina virtual, abra su navegador preferido y escriba en la barra de direcciones lo siguiente: http://IP_máquina_virtual:8069

Una vez hecho todo esto ya se puede crear la base de datos y realizar las funciones deseadas en Odoo.

A.3 Instalacion Atom

Atom ha sido el editor de código utilizado para la realización de este proyecto, el cual ha facilitado mucho la programación tanto de Python como de XML. La forma de instalarlo en Ubuntu es la siguiente:

Primero se introduce la siguiente línea en la terminal:

```
#sudo apt-get install build-essential git libsecret-1-dev fakeroot rpm libx11-dev libxkbfile-dev
```

Una vez hecho esto introducimos en la terminal:

```
#git clone https://github.com/atom/atom.git
#cd atom
#script/build
```

Tras esto Atom ya estará instalado en nuestro Ubuntu.