



UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA



UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Informática

Desarrollo de un módulo de OdoO para el análisis de los
datos de una academia

Trabajo Fin de Grado

Grado en Ingeniería Informática

AUTOR/A: Navalón Arnal, Pablo

Tutor/a: Busquets Mataix, José Vicente

Cotutor/a: Pérez Blasco, Pascual

CURSO ACADÉMICO: 2023/2024

Resumen

Este proyecto consiste en el desarrollo de un módulo para el análisis de datos de una academia. Para afrontar esta problemática, se utiliza la tecnología ERP (Enterprise Resource Planning), que se trata de un software integrado y modular que permite el desarrollo de soluciones genéricas o específicas para las empresas de todo tipo de sectores. De entre la gran variedad de ERP, Odoo ha sido la elección dada su flexibilidad, accesibilidad y personalización, además de ser software libre.

Para el desarrollo del módulo se utiliza la herencia de clases, herramienta importante para modificar las funcionalidades de los módulos básicos y adaptarlas a las necesidades del proyecto. La población de datos también se ha abordado durante el desarrollo, permitiendo contar con muestras de datos útiles previas a la puesta en marcha real del proyecto.

Se aborda el análisis de datos como temática principal. Para ello se introducen las herramientas de análisis de datos, siendo Grafana la plataforma utilizada para la representación gráfica de los datos.

La suma de todas estas tecnologías nos deja con un software completo que cubre las necesidades de una academia de repaso.

Palabras clave: ERP, Odoo, módulo, análisis de datos, Grafana

Abstract

This project involves the development of a module for data analysis for an academy. To address this problem, ERP (Enterprise Resource Planning) technology is used, which is an integrated and modular software that allows the development of generic or specific solutions for companies in all types of sectors. Among the wide variety of ERPs, Odoo has been the choice given its flexibility, accessibility and customization, as well as being free software.

For the development of the module, class inheritance is used, an important tool for modifying the functionalities of the basic modules and adapting them to the needs of the project. The data population has also been addressed during development, allowing for useful data samples prior to the actual implementation of the project.

Data analysis is addressed as the main theme. For this purpose, data analysis tools are introduced, with Grafana being the platform used for the graphical representation of the data.

The sum of all these technologies leaves us with a complete software that covers the needs of a tutoring academy.

Keywords: ERP, Odoo, module, data analysis, Grafana

Índice de contenidos

1. Introducción.....	7
1.1. Motivación	7
1.2. Objetivos	7
1.3. Estructura de la memoria	8
2. Estado del arte.....	9
2.1. ERP	9
2.2. Odoo.....	12
2.3. Análisis de datos:	15
2.4. Comparativa entre plataformas	19
2.5. Elección de la herramienta	20
3. Análisis del problema.....	21
3.1. Especificación de requisitos del software	21
3.2. Casos de uso.....	30
3.3. Identificación y análisis de soluciones posibles.....	34
4. Diseño de la solución	36
4.1. Arquitectura del sistema.....	36
4.2. Módulos de Odoo que componen la aplicación	37
4.3. Población de los modelos.....	37
4.4. Prototipos.....	38
5. Desarrollo de la solución.....	40
5.1. Introducción al desarrollo	40
5.2. Directorio del proyecto	40
5.3. Herencia.....	42
5.4. Models.....	44
5.5. Views.....	48
5.6. Consultas SQL.....	50
6. Implantación	56
6.1. Configuración de la máquina virtual y las plataformas Odoo y Grafana.....	56
7. Pruebas.....	60
8. Conclusiones	67
Bibliografía	69
Anexos - Glosario.....	70
Anexos – Objetivos de desarrollo sostenible (ODS).....	71

Índice de tablas

Tabla 1: Evolución de Odoo. Fuente: otros autores (9)	13
Tabla 2: Modelo	23
Tabla 3: Iniciar sesión administrador Odoo	24
Tabla 4: Cerrar sesión administrador Odoo	24
Tabla 5: Crear contacto	24
Tabla 6: Eliminar contacto	25
Tabla 7: Visualizar contacto.....	25
Tabla 8: Editar contacto	25
Tabla 9: Crear servicio	26
Tabla 10: Eliminar servicio.....	26
Tabla 11: Visualizar servicio.....	26
Tabla 12: Visualizar servicio.....	26
Tabla 13: Facturar servicio.....	27
Tabla 14: Gestionar eventos en el calendario.....	27
Tabla 15: Iniciar sesión administrador Grafana	27
Tabla 16: Cerrar sesión administrador grafana	28
Tabla 17: Crear gráfico	28
Tabla 18: Visualizar gráfico.....	28
Tabla 19: Iniciar sesión profesor.....	29
Tabla 20: Cerrar sesión profesor	29
Tabla 21: Gestionar eventos en el calendario como profesor.....	29
Tabla 22: Visualizar contacto como profesor.....	30

Índice de ilustraciones

Ilustración 1: Modelo Vista Controlador. Fuente: otros autores	15
Ilustración 2: Caso de uso general.....	31
Ilustración 3: Caso de uso contactos	32
Ilustración 4: Caso de uso servicios	32
Ilustración 5: Caso de uso Grafana	33
Ilustración 6: Caso de uso profesores.....	33
Ilustración 7: Arquitectura del sistema.....	36
Ilustración 8: Pototipo Grafana	38
Ilustración 9: Directorio del proyecto	40
Ilustración 10: Manifest.	41
Ilustración 11: Herencia de modelos en Odoo	43
Ilustración 12: Herencia de vistas	43
Ilustración 13: Especificaciones de herencia.....	44
Ilustración 14: __init__.py	45
Ilustración 15: res_partner.....	45
Ilustración 16: Menus.xml.....	48
Ilustración 17: Res_partner_views.xml.....	49
Ilustración 18: Contacts_views.xml	49
Ilustración 19: Categorías de asignaturas más populares	52
Ilustración 20: Ventas totales por asignatura	52



Desarrollo de un módulo del ERP Odoo para el análisis de datos de una academia

Ilustración 21: Ganancia anuales.....	53
Ilustración 22: Ganancias mensuales	53
Ilustración 23: Alumnos mensuales	54
Ilustración 24: Número de alumnos por edades	55
Ilustración 25: Dashboard academia	55
Ilustración 26: Intalación del módulo.....	58
Ilustración 27: Fin instalación	58
Ilustración 28: Comando para la población de la base de datos.....	59
Ilustración 29: Resultado de la población	59
Ilustración 30: Iniciar sesión Odoo	60
Ilustración 31: Vista inicial	60
Ilustración 32: Acceso menú contactos	61
Ilustración 33: Menú contactos	61
Ilustración 34: Formulario contacto	61
Ilustración 35: Suprimir	61
Ilustración 36: Acceso menu servicios.....	62
Ilustración 37: Menú servicios	62
Ilustración 38: Formulario servicios	62
Ilustración 39: Acceso menu facturas	63
Ilustración 40: Menú facturas.....	63
Ilustración 41: Formulario facturas	63
Ilustración 42: Acceso menu calendario	63
Ilustración 43: Formulario calendario	64
Ilustración 44: Cerrar sesión Odoo.....	64
Ilustración 45: Inicio sesión Grafana	65
Ilustración 46: Menú inicial	65
Ilustración 47: Menú dashboards	65
Ilustración 48: Cerrar sesión Grafana.....	66

1. Introducción

Vivimos en la era de la información, las empresas afrontan diariamente el reto de trabajar con millones de bytes de información. Aprender a manejarlos se ha convertido en algo imperativo para poder mejorar la gestión de todos los procesos de una organización. Es por ello que la integración de software como los ERP (Enterprise Resource Planning) en las empresas se está convirtiendo en una obligación. Esta tecnología ayuda a optimizar la toma de decisiones, la organización interna, la conexión entre departamentos y proporciona toda la información necesaria para mejorar la productividad y el crecimiento a futuro. Se trata de una herramienta para el manejo de la información.

Para entender la utilidad y los retos que supone la aplicación de los ERP Odoo, analizaremos su uso en el caso de una academia.

A continuación abordaremos la motivación, los objetivos y la estructura de la memoria

1.1. Motivación

Después de relizar prácticas en empresas y entrar en contacto con los sistemas ERP, me ha llamado la atención lo indispensables que se han vuelto hoy en día para el manejo de los procesos en las organizaciones. Es por ello que participar en el desarrollo de uno de ellos que ayude a mejorar la gestión de las empresas pequeñas, como puede ser una academia, me parece interesante a la vez que necesario. También la oportunidad de aprender más sobre ellos y conocer un tipo de ERP de código abierto como es Odoo lo hace más llamativo todavía.

1.2. Objetivos

El objetivo de este proyecto es el desarrollo de un módulo de análisis de datos para el ERP Odoo. Esta aplicación estaría dirigida a centros académicos de repaso, proporcionando una mejor gestión de los recursos gracias a un mayor conocimiento del entorno.

Se plantearán a continuación los objetivos necesarios para poder llevar a cabo el desarrollo del módulo. Gracias a él la academia podrá recopilar y analizar sus datos, lo que facilitará su toma de decisiones.

- Adquirir información sobre el software ERP
- Aprendizaje del ERP Odoo, adquiriendo capacidades de desarrollador
- Aprender sobre análisis de datos y sus herramientas
- Instalación y configuración de Odoo y otras posibles herramientas
- Desarrollo de un módulo funcional

1.3. Estructura de la memoria

- **Introducción:** se plantean los apartados de la motivación, los objetivos y la actual estructura de la memoria.
- **Estado del arte:** se definen los ERP, su historia, los tipos de ERP, sus objetivos y características y algunos de los ERP más populares. Se define Odoo y su propia historia, se habla de sus características y de su funcionamiento. Posteriormente se define el análisis de datos, ETL, qué son las herramientas de análisis de datos, exposición de distintas herramientas y valoración de cada una de ellas. Conclusión y ejecución de la herramienta Grafana.
- **Análisis del problema:** especificación de requisitos del software siguiendo el estándar IEEE830, requisitos específicos, casos de uso e identificación y análisis de soluciones posibles.
- **Diseño de la solución:** se explica la arquitectura del sistema, los módulos que se utilizan, la población de datos y el prototipo.
- **Desarrollo de la solución propuesta:** se expone el directorio del proyecto, qué es la herencia, los modelos que conforma la solución, las vistas y las consultas sql
- **Implantación:** se describe como configurar el entorno de la máquina virtual y las plataformas utilizadas.
- **Pruebas funcionales:** se muestran las funcionalidades del módulo y se valora el cumplimiento de los requisitos.
- **Conclusiones:** se valora el cumplimiento de los objetivos propuestos, los problemas encontrados y sus soluciones y la relación con los estudios y aprendizajes.
- **ODS:** Se relacionan los Objetivos de Desarrollo Sostenible con el proyecto.
- **Glosario:** Definición de términos relacionados con el proyecto.

2. Estado del arte

Este apartado hablará de las tecnologías utilizadas para el desarrollo del proyecto, el software ERP y las herramientas de análisis de datos. Se comenzará definiendo qué es un ERP, se nombrarán sus características principales y su desarrollo a lo largo de la historia hasta la actualidad. Se nombrarán y expondrán algunos de los ERP más populares y se profundizará en la arquitectura de Odoo además de hablar de su historia. Por último, se hablará de la herramienta de análisis de datos Grafana.

2.1. ERP

Para comprender lo que es un ERP, primero debemos plantear algunos conceptos previos. Forman parte de una familia de *software* llamados sistemas de gestión, entre los que se encuentran también productos como los CRM (*Customer Relationship Management*) o los BI (*Business Intelligence*). Estos sistemas de gestión se encargan de proveer a las organizaciones de herramientas con las que gestionar, organizar e integrar mejor todos los procesos de negocio en un mismo entorno. Los ERP entran dentro de esta categoría de herramientas y se expondrán a continuación.

2.1.1. Definición principal ERP

ERP o *Enterprise Resource Planning* es un sistema de gestión integrado de planificación de recursos empresariales. Son sistemas centrados en la gestión de procesos empresariales que buscan la optimización y automatización de estos mediante el manejo de la información.

Es un *software* de apoyo a las organizaciones que cuenta con características como adaptabilidad, flexibilidad y modularidad, cuyo principal objetivo es facilitar a las organizaciones tener un mayor grado de control sobre todas sus áreas de negocio (1).

2.1.2. Historia del ERP

Es evidente que este software no surgió tal y como es actualmente, es por ello que vamos a repasar su mejoría y desarrollo a lo largo de los años a continuación.

Sus inicios datan a finales de la segunda guerra mundial (1940-1950), utilizándose programas para organizar temas de logística y producción militar.

Más adelante, en la década de los años 60 se incorporan al mercado las primeras computadoras comerciales dirigidas a empresas para la gestión de su información. En un primer momento el software venía integrado con el propio hardware, lo que evolucionó poco a poco hacia la producción independiente de software a medida. Surgen aquí programas como BOM (Listas de materiales) y IMC (Gestión de inventario), además de las primeras empresas centradas en este tipo de desarrollo de software.

Desarrollo de un módulo del ERP Odoo para el análisis de datos de una academia

Ya en la década de los setenta comienza la planificación de las necesidades de los materiales, es decir, se plantea el cómo, dónde, cuándo y cuánto material se iba a usar. Aparecen en esta década la mayoría de empresas proveedoras de ERP, como SAP.

Así pues, en los ochenta surge la Planificación de recursos de la producción (MRP_II), se añaden a las necesidades anteriores los costes de adquisición, logística y mano de obra.

Finalmente, en la década de los noventa se le acuña con el nombre “ERP”, lo que anteriormente era una herramienta de planificación, pasa a ser un sistema de información fundamental para la toma de decisiones en cualquier empresa

A partir de los 2000, empieza su expansión con funciones propias de otros programas como SCM (gestión de cadena de suministro) o CRM (gestión de los clientes). Después surgen ERP orientados hacia software en la nube, lo cual elimina los problemas y costes de instalación y configuración. También se añaden herramientas como Business Intelligence (BI) y aparecen ERP de código abierto como Odoo (2).

Tipos ERP (3)

Dada la gran variedad de organizaciones existentes, ha surgido distintos tipos de ERP para adaptarse a las necesidades de cada una.

Por lo general se dividen en las siguientes categorías.

Según su concepción:

- **Genéricos:** son ERP con características generales para poder ser utilizados por empresas de todo tipo de sectores.
- **Preparametrizados:** son ERP con características propias de cierto sector o adaptadas a cierto tamaño de empresa.
- **Individualizados:** son ERP diseñados a medida para cubrir las necesidades específicas de cada empresa.

Según el tipo de instalación:

- **On premise:** o instalación en local es el tipo de instalación que más autonomía ofrece a las empresas. El software se alberga en las propias máquinas de las organizaciones y estas se encargan de gestionar enteramente su funcionamiento, teniendo todo el control sobre las actualizaciones, mantenimiento, almacenamiento y seguridad del sistema. Suele ser la instalación más común en las grandes empresas, que cuentan con la economía y los recursos necesarios para su gestión adecuada. Se requiere una gran inversión inicial.
- **ERP Cloud:** o instalación en la nube es actualmente el tipo de instalación más ofertado por los desarrolladores de ERP y generalmente el más cómodo para las empresas. Parte del funcionamiento como es el almacenamiento, mantenimiento, actualizaciones y seguridad quedan a cargo del proveedor. Delegando gran parte del control sobre el sistema a terceros obteniendo a cambio comodidad. No es necesaria una gran inversión económica dado que no requiere gastos en ampliaciones ni mejoras de los propios sistemas informáticos, sin embargo se requerirá de una inversión mensual o anual por el pago de una suscripción.

- **Software** como servicio: proviene de las siglas en inglés SaaS (*Software as a Service*). Se trata de otra versión de instalación en la nube, cambiando el modelo de negocio para ofrecer un ERP como un servicio *software*. La empresa no necesita hacerse cargo del sistema, simplemente con contar con un navegador *web* ya es suficiente. Todo el funcionamiento es delegado a terceros y los costes pasan a ser suscripciones mensuales o anuales como en el modelo *Cloud*.

2.1.3. Objetivos y características de un ERP

Se explicaran sus rasgos principales que lo definen como la clase de software que es y sus características medibles.

Sus rasgos principales son:

- **Prefabricado:** se construye de forma que sea compatible con la mayoría de áreas de negocio.
- **Integrado:** trata de conectar los procesos de negocio de la organización entre sí.
- **Modulares:** están diseñados de tal manera que sus funcionalidades se pueden dividir en paquetes, es decir, se añaden o quitan funciones en base a las necesidades de la organización que requiera el software.
- **Adaptable:** ligada al rasgo anterior, gracias a su modularidad los ERP consiguen adaptarse al entorno modificando su funcionamiento.

Existen también una serie de atributos o características medibles que aseguran que los ERP puedan cumplir con los requerimientos de negocio:

- **Funcionalidad:** es la capacidad del *software* para resolver adecuadamente las necesidades del usuario.
- **Usabilidad:** es la facilidad de uso del sistema.
- **Facilidad de mantenimiento:** capacidad de modificarlo a lo largo del tiempo.
- **Portabilidad y compatibilidad:** capacidad para ejecutarse en diferentes entornos y sistemas operativos.
- **Complejidad y modularidad:** la habilidad para implementar módulos adecuados a las necesidades del cliente y la interacción entre estos.
- **Reusabilidad:** capacidad de reutilización del código *software*.

2.1.4. Principales ERP (4)

Se comentarán algunos de los ERP más utilizados del mercado para obtener una perspectiva general de del mercado

- **SAP S/4HANA.** Se trata de uno de los ERP más consolidados y robustos del mercado, utilizado generalmente por grandes y medianas empresas. Está montado sobre la base de datos HANA, la cual está optimizada para el procesamiento y análisis de datos en tiempo real. Destaca por ofrecer una alta personalización específica a cada empresa, adaptándose a demanda a los requerimientos de cada una mediante el uso de tecnologías avanzadas como *IA* y *Machine Learning*.



- **Microsoft Dynamics 365.** Es el ERP insignia de Microsoft, sus puntos fuertes son la compatibilidad con una gran gama de productos de Microsoft, como pueden ser *Office 365* o *PowerBI*. Está pensado para empresas de tamaño medio ofreciendo gran cantidad de módulos flexibles así como *CRM*. También utiliza las tecnologías *IA* y *Machine Learning*.
- **Oracle ERP Cloud:** Es un ERP basado exclusivamente en la nube, ahorrando todos los costes de instalación y ofreciendo capacidades innovadoras como su infraestructura de segunda generación y su constante actualización de funcionalidades. Destaca en por su escalabilidad y velocidad de procesamiento en módulos de finanzas y cadena de suministro
- **Sage X3:** *Software* enfocado en las pequeñas y medianas empresas, destacando en las finanzas, la producción y la cadena de suministro. Es bastante personalizable y robusto, aunque su escalabilidad y alcance están limitados si se compara con los grandes ERP del mercado.
- **NetSuite:** Se trata de un ERP genérico ideal para empresas en crecimiento. Se basa en ofrecer una solución integral formada por el ERP, *CRM* y *e-commerce*. No es demasiado flexible pero se implementa rápido y es fácil de utilizar.

2.2. Odoo

2.2.1. Historia de Odoo (8)

El creador de Odoo fue Fabien Pinkaers, comenzando a desarrollar TinyERP en el año 2005. Por aquel momento, SAP ya era considerado el ERP por excelencia, por lo que Pinkaers se propuso diseñar el mejor ERP de software libre posible para ser competitivo en el mercado. Unos años más tarde renombró su software para llamarlo OpenERP, y la empresa consiguió estabilidad en el mercado. En este punto se centraron en expandir su red de socios, lo que mejoró su situación y consiguieron convertirse en el software de gestión más instalado del momento. Finalmente en 2014 cambiaron el nombre al actual Odoo. Seguidamente, repasaremos las distintas versiones lanzadas desde el inicio hasta ahora con esta tabla

Tabla 1.

Nombre programa	Versión	Fecha lanzamiento	Tipo software
Tiny ERP	1.0	Febrero de 2005	GNU GPL
	2.0	Mayo de 2005	GNU GPL
	3.0	Septiembre 2005	GNU GPL
	4.0	Diciembre de 2006	GNU GPL
OpenERP	5.0	Abril de 2009	GNU GPL
	6.0	Enero de 2011	GNU AGPL
	6.1	Febrero de 2012	GNU AGPL
	7.0	22 de diciembre de 2012	GNU AGPL
Odoo	8.0	18 de septiembre de 2014	GNU AGPL
	9.0	1 de octubre de 2015	GNU LGPL v3/ Odoo Enterprise Edition License v1.0
	10.0	5 de octubre de 2016	GNU LGPL v3/ Odoo Enterprise Edition License v1.0
	11.0	5 de octubre de 2017	GNU LGPL v3/ Odoo Enterprise Edition License v1.0
	12.0	3 de octubre de 2018	GNU LGPL v3/ Odoo Enterprise Edition License v1.0
	13.0	5 de octubre de 2019	GNU LGPL v3/ Odoo Enterprise Edition License v1.0
	14.0	5 de octubre de 2020	GNU LGPL v3/ Odoo Enterprise Edition License v1.0
	15.0	3 de octubre de 2021	GNU LGPL v3/ Odoo Enterprise Edition License v1.0
	16.0	12 de octubre de 2022	GNU LGPL v3/ Odoo Enterprise Edition License v1.0

Tabla 1: Evolución de Odoo. Fuente: otros autores (9)

En la tabla vemos que inicialmente el software se distribuyó con el nombre de TinyERP (2005 a 2009), después con el nombre OpenERP (2009 a 2014) y finalmente en el 2014 se renombra a Odoo y en 2015 se lanzaron dos versiones del producto con dos licencias distintas (representadas en la misma fila de la

Tabla 1).



2.2.2. Características

Odoo es un *software* de código abierto que ofrece un servicio ERP basado en la gestión de procesos. Está dividido en módulos como por ejemplo ventas, recursos humanos o sitio web, además de ofrecer servicios *CRM*.

Actualmente y desde 2015 Odoo dispone de dos versiones. Contamos con la versión *Community edition*, la cual es gratuita y ofrece libertad de uso y personalización ya que cuenta con la licencia LGPLv3 (*GNU Lesser General Public License*), mientras que por otro lado tenemos la versión *Enterprise edition*, la cual es de pago y ofrece todas las características de la versión gratuita más funcionalidades extra como pueden ser el asesoramiento para el desarrollo de módulos específicamente adaptados para el cliente.. Tanto la versión *Community* como la *Enterprise* cuentan con distintos tipos de instalación: en local, *Cloud* o mediante la plataforma Odoo.sh.

Las características modulares y flexibles de los ERP se trasladan también a Odoo, dada la cantidad de módulos ofrecidos por la propia empresa además de la inmensa cantidad que ha desarrollado la comunidad. El usuario puede decidir en todo momento qué módulos instalar y cuáles no, simplemente ha de tener en cuenta las dependencias propias de cada uno.

Dada la arquitectura de Odoo, que explicaremos en el siguiente apartado, Odoo utiliza tres tecnologías para diseñar su software. Para el desarrollo de la lógica de sus aplicaciones, utiliza Python como lenguaje de programación. Python es un lenguaje de alto nivel orientado a objetos multiparadigma basado en la programación imperativa. Para la representación visual de la aplicación, o como lo llama Odoo, de las vistas, utilizan el lenguaje XML (*eXtensible Markup Language*), un metalenguaje que define las reglas de codificación de los documentos, desarrollado por W3C (*World Wide Web Consortium*). La última de las tecnologías está dedicada al almacenamiento de los datos. Se trata de PostgreSQL como lenguaje de gestión de bases de datos relacionales, basado en SQL. Cobrará especial importancia en nuestro desarrollo.

Pasamos ahora con la explicación de su funcionamiento y arquitectura.

2.2.3. Funcionamiento

Odoo utiliza la arquitectura multinivel Modelo Vista Controlador (MVC), el cual divide el funcionamiento de la aplicación en tres capas, como se explica en la **Ilustración 1**.

- **Capa de datos:** la información se almacena y extrae de la base de datos PostgreSQL. Se comunica solamente con la capa lógica.
- **Capa lógica:** se encarga de coordinar la aplicación mediante el procesamiento de los datos. Comunica la capa de datos con la capa de presentación. El lenguaje para la lógica en Odoo es Python.
- **Capa de presentación:** se encarga de traducir los resultados que le transmite la capa lógica al usuario de una manera visual. Está escrita en XML

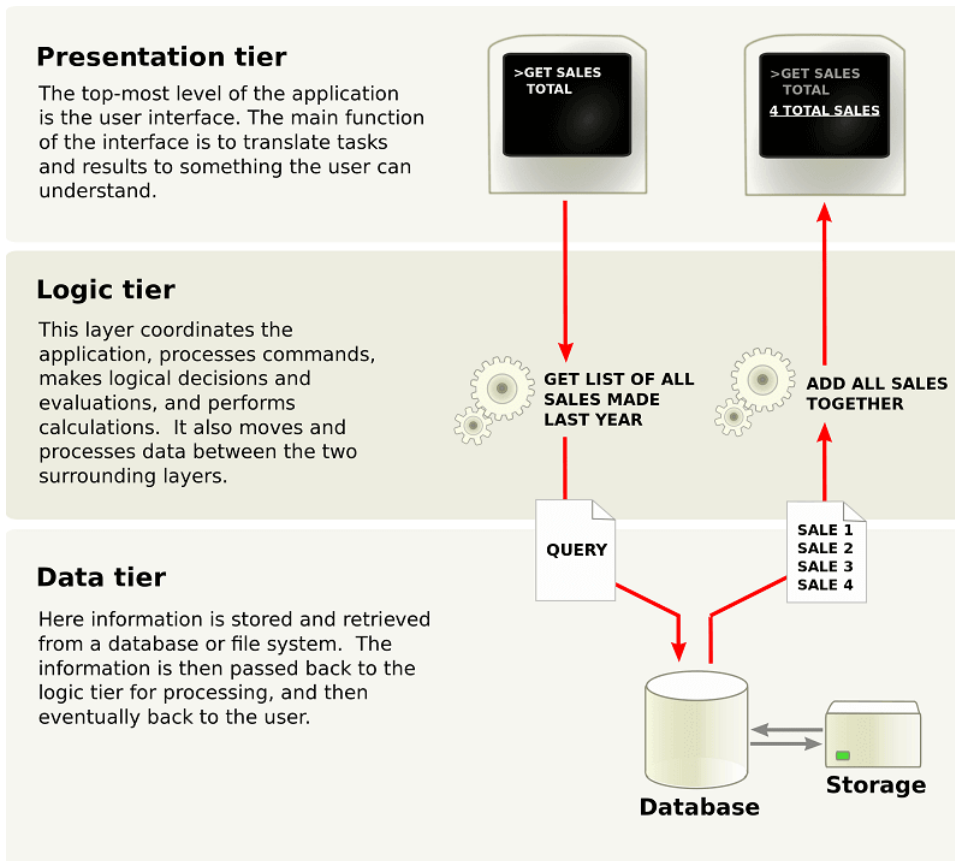


Ilustración 1: Modelo Vista Controlador. Fuente: otros autores

2.3. Análisis de datos:

Una vez elegido el ERP en el que desarrollaremos nuestro módulo, debemos plantearnos qué es el análisis de datos y que herramientas utilizaremos para analizar los datos gestionados dentro del módulo.

El análisis de datos es una disciplina que se centra en convertir datos sin procesar en información práctica que facilite la toma de decisiones. Para ello se utilizan herramientas, tecnologías y procesos para solucionar problemas mediante la observación de los datos. Procesos tales como la exploración, transformación y examinación de datos y herramientas o plataformas externas que se especializan en la exploración de datos y el examen de estos.

El primer paso para el análisis de datos suele ser llevar a cabo una ETL (5).

2.3.1. ETL

ETL hace referencia a Extract, Transform and Load o Extracción, Transformación y Carga de datos en español. Esto permite a las organizaciones el formateo, limpieza y carga de datos provenientes de diversas fuentes en su sistema, lo que supone una gran ventaja en, por ejemplo, situaciones de migración de sistemas

- **Extracción.** Consiste en extraer todos los datos de la fuente o fuentes de origen. Cada fuente puede proveer los datos en un formato diferente, normalmente de bases de datos relacionales o ficheros planos. En la extracción se preparan los datos en un formato que permita su transformación, a la vez que se rechazan los que no lo cumplan. Su impacto en la fuente de origen durante el proceso debería ser el mínimo posible, para poder mantener su normal funcionamiento
- **Transformación.** Transforma los datos aplicando una serie de funciones o reglas de negocio para convertirlos a un formato compatible con el sistema destino. Algunas transformaciones consisten en eliminar valores nulos o traducir códigos.
- **Carga.** Los datos preparados son cargados en el sistema destino.

2.3.2. Herramientas de análisis de datos

Una vez se tienen los datos en el sistema deseado, ya se puede hacer uso de una herramienta o plataforma de análisis de datos. Una herramienta de análisis de datos permite al usuario procesar, analizar y presentar los datos de forma comprensible.

Disponemos de dos opciones para nuestro proyecto, utilizar las herramientas proporcionadas por el propio Odoo y desarrollar un panel de datos o “dashboard” dentro del módulo o utilizar una herramienta de análisis de datos externa.

2.3.3. Dashboard Odoo

Odoo proporciona una manera de desarrollar módulos para el análisis de datos que permite la creación de gráficos y visualización de datos en estos. Odoo al ser un ERP más enfocado en cubrir todos los procesos de negocio, no dispone de capacidades sobresalientes a la hora de cubrir la disciplina de análisis de datos. Sin embargo, cuenta con algunas formas para cumplir la tarea.

Odoo cuenta con dos maneras de crear reports personalizados, sobrescribiendo el método “BaseModel.init()” o usando “_table_query”. Independientemente del método utilizado, se ejecutará una sentencia SQL para recuperar la información de la base de datos y representarla en los gráficos.

Para crear los paneles con los gráficos que permitirán visualizar los datos, contamos con una estructura de datos que se compone de las vistas Graph, Pivot y Dashboard. Estas vistas se personalizan mediante código xml. Graph y Pivot se utilizan para visualizar agregaciones de los registros mediante cuatro tipos de gráficos: de barras, circular, de líneas y pivote. Dashboard es una estructura de datos más compleja que permite la creación de subvistas,

incluyendo las vistas posibles creadas en Graph y Pivot, pero con más opciones personalizables, como la adición de dominios o de fórmulas. Se entiende como un lienzo que puede albergar distintos gráficos en sí mismo, mientras que Graph y Pivot utilizan toda la ventana para un único gráfico.

La vista Dashboard, sin embargo, se trata de una característica del paquete Enterprise, por lo que para el desarrollo libre que estamos realizando solo contaríamos con Graph y Pivot. Esto merma enormemente nuestra capacidad de desarrollo de un módulo de análisis de datos, por lo que se ha decidido prescindir de las opciones proporcionadas por Odoon y valorar herramientas externas.

Las herramientas externas que se van a explicar son: Metabase, Superset y Grafana.

2.3.4. Metabase

Plataforma de análisis y visualización de datos de código abierto diseñada para facilitar el análisis de datos. Su principal baza es la rapidez con la que te permite cargar datos y construir tu primer dashboard para su análisis, así como su enfoque en facilitar su uso a usuarios de todos los niveles, sin necesidad de que sean expertos en análisis de datos. También permite la integración de datos de múltiples fuentes SQL, lo que lo hace todavía más accesible. Su interfaz es clara y representa los datos de forma sencilla e intuitiva, permitiendo filtrar, resumir u ordenar entre otros. Permite compartir los resultados obtenidos de forma que todo el mundo puede visualizar y explorar los *dashboards* creados, ya sea con acceso directo mediante links o publicados en una web. A su vez, se pueden crear alertas que avisan cuando se cumple cierta condición a demanda. Cuenta también con analíticas embebidas.

Entre sus principales características están:

- **Soporte SQL.** Cuenta con la posibilidad de integrar múltiples fuentes de datos SQL.
- **Query Builder.** Es una herramienta de “*point-and-click*” que permite la creación de sentencias *SQL* sin necesidad de conocimientos técnicos. De forma visual se puede filtrar, utilizar *Joins*, expresiones personalizadas, agregación, etc. También dispone de un editor *SQL* para usuarios avanzados.
- **Drill-through.** Permite navegar por los *dashboards* de forma rápida haciendo click en los gráficos, en los modelos y demás elementos para visualizarlos con mayor detalle.
- **Dashboards analíticos.** Diseñados específicamente para el análisis de datos, con la posibilidad de aplicarles filtros, hacer “*drill-through*”, clicks personalizados, carga rápida, etc
- **Controles de acceso.** Restricciones a la hora de quien puede y quien no puede visualizar determinados *dashboards* o incluso columnas/filas de datos de una tabla.
- **Modelos.** Creación de modelos de datos personalizados
- **Carga de CSV.** Permite la carga y análisis directo de datos al cargar archivos CSV

Su uso se puede dar de dos formas:

- **Instalación en local.** El cliente se gestiona su propio mantenimiento y seguridad



- **Metabase Cloud.** Metabase se encarga de proporcionar toda la infraestructura para que el usuario no se tenga que preocupar de nada. Actualizaciones, backups, seguridad de los datos, alertas y soporte continuo.

2.3.5. Superset

Apache Superset es una aplicación web open source de exploración y visualización de datos. Está diseñada para ser accesible a personas tanto avanzadas como principiantes. Tiene gráficos que van desde simples a complejos.

Sus características principales son:

- **Interfaz intuitiva y de fácil uso.** Cuenta con una gran variedad de personalización ya preinstalada y/o diseñable.
- **Explore.** Constructor de dashboards o vistas sin necesidad de código. También dispone de SQL Lab, que permite la introducción de sentencias SQL para mayor complejidad.
- **Advanced Analytics.** Herramienta para la transformación de datos que permite definir métricas y dimensiones.
- **Soporte SQL.** Soporta muchos tipos de bases de datos e integración con distintos backends.
- **Plugins.** Permite la creación de *plugins* para personalizar las visualizaciones y cuenta con una *API* para personalizar la programación.

Sólo está disponible como aplicación en la nube por lo que está diseñada para ser escalable. Permite elegir entre Web server, metadatos, message queue, results backend y caching layer.

2.3.6. Grafana

Es una plataforma de visualización de datos de código abierto que permite la creación de dashboards dinámicos y el análisis de datos.

Está pensado para la observabilidad de datos en tiempo real, lo que permite monitorear activamente cualquier problema que pueda surgir.

Grafana se define como una herramienta que forma parte del proceso de DevOps para optimizar el proceso de entrega de software. Concretamente en las fases de operar y monitorear el software. La información que utiliza grafana principalmente se basa en métricas, logs y trazas.

Sus principales bazas son:

- Gran compatibilidad con diferentes tipos de bases de datos y sistemas de monitorización, así como su capacidad de integración con servicios en la nube y otras herramientas. Esto permite al usuario consolidar datos de diferentes fuentes en un mismo panel de control.
- Su flexibilidad y personalización. Ofrece tanto paneles prefabricados como personalizados al gusto del usuario, con muchos tipos de gráficos y opciones de visualización.

- Interfaz intuitiva y fácil de utilizar a la vez que escalable en complejidad. Es accesible a usuarios nuevos a la vez que ofrece una extensa funcionalidad con capacidades avanzadas para usuarios expertos.
- Soporte para métricas, logs y trazas. Permite tener una observabilidad más completa de los sistemas.
- Comunidad activa y ecosistema de *plugins*. La comunidad de Grafana es muy activa y está constantemente desarrollando *plugins* y extensiones para extender su funcionalidad. Los usuarios se pueden beneficiar de ello al disponer de soluciones más específicas para su caso.

Se puede utilizar desde varias plataformas:

- **Grafana Open Source.** instalación en Local y uso propio.
- **Grafana Cloud.** Se utiliza en la nube.
- **Grafana Enterprise.** Para empresas con requerimientos especiales

2.4. Comparativa entre plataformas

A la hora de elegir la plataforma de análisis y visualización de datos, hay varios factores a tener en cuenta. Vamos a ir uno por uno:

1. **Facilidad de uso.** A la hora de aprender una herramienta nueva, hay que tener en cuenta cuan compleja es de aprender. Grafana ofrece una interfaz intuitiva y fácil de usar, sin embargo, tiene una curva de aprendizaje más pronunciada. Aun así esto no debería suponer un problema ya que dispone de una amplia documentación. Mientras tanto, Metadata está preparada para instalarse y configurarse rápidamente, para ser accesible a todo tipo de usuarios. Superset por su parte es similar a Metadata en términos de facilidad de uso, pero no es tan accesible de primeras.
2. **Flexibilidad y personalización.** Grafana tiene una gran variedad de opciones de personalización de vistas y paneles, pudiendo cambiar su apariencia y comportamiento a gusto del usuario. Además, cuenta con infinidad de plugins que permiten extender su funcionamiento. Superset también cuenta con muchas opciones de personalización de vistas, aunque no dispone de tantas ni tantos plugins como Grafana. Metadata es la plataforma más limitada en este sentido, aunque no por mucha diferencia.
3. **Compatibilidad con fuentes de datos.** Este es el punto fuerte de Grafana, dada su popularidad en el campo de análisis de datos, dispone de una cantidad desmedida de compatibilidad con fuentes de datos, pudiendo incluso desarrollar un plugin si no existe la conexión. También está bien preparada para crear paneles que provengan de varias fuentes de datos. Superset y Metadata por su parte cuentan con una buena cantidad de fuentes de datos, pero no son comparables a las de Grafana.
4. **Capacidad de análisis y exploración de datos.** Aquí tanto Grafana como Superset ofrecen una experiencia similar avanzada en análisis de datos, con opciones incluso de transformación de los datos. Metadata ofrece una experiencia más básica, pero suficiente en muchos casos.



5. **Seguridad y control de acceso.** Todas las plataformas tienen controles de seguridad adecuados, a nivel de paneles, roles y usuarios. Sin embargo, si se requiere de una seguridad más exhaustiva, Grafana y Superset tienen muchas más, llegando a autenticación de usuarios, control basado en roles y permisos granulares a nivel de panel y a nivel de columna entre otros. Superset está más centrado en seguridad empresarial.
6. **Soporte y comunidad.** Las tres plataformas tienen una comunidad activa con usuarios que proveen desarrollos nuevos, pero la más completa con diferencia es la de Grafana. Cuenta con muchos desarrolladores en todo el mundo y muchos usuarios activos que crean constantemente nuevos plugins e integraciones, además de una gran variedad de tutoriales y un foro donde acudir en caso de ayuda.

2.5. Elección de la herramienta

Una vez analizadas las capacidades de las tres herramientas y puestas en perspectiva, las tres cumplen con los requerimientos necesarios para poder desarrollar nuestro módulo de Odoo en conjunto. Se tratará más bien de una elección en base a la preferencia. La elección será Grafana.

Respecto a la facilidad de uso de las herramientas, la curva de aprendizaje no debería ser un problema para nosotros, ya que contamos con el apoyo de la documentación y el tiempo disponible a dedicar al aprendizaje. Es por ello que Grafana aun no siendo tan favorable con los nuevos usuarios como Metabase, sigue siendo la mejor opción. Las fuentes de datos no tendrán peso especial en la elección de la plataforma, ya que Odoo solamente utiliza PostgreSQL y todas ellas son compatibles con esta. A pesar de todo, tiene cierto peso en caso de futuros desarrollos que requieran de otras fuentes. La flexibilidad y personalización que ofrecen todas las plataformas resultan más que suficientes en nuestro caso, pero puestos a elegir, Grafana es la que cuenta con más variedad. Esto también sucede en la parte técnica a la hora de analizar y explorar los datos y en la parte de seguridad y control de acceso, donde Grafana sigue siendo superior. En lo personal, lo que marca realmente la diferencia es la comunidad. Grafana cuenta con mucha más ayuda online gracias a los tutoriales, la documentación, los desarrollos ya realizados y los plugins disponibles que pueden ayudarnos a la resolución de futuros problemas.

3. Análisis del problema

El análisis del problema se pretende plantear mediante la Especificación de requisitos *software* (ERS), el cual sigue el estándar IEEE 830. Se trata de una guía que propone una serie de recomendaciones que cumplir a la hora de desarrollar software.

3.1. Especificación de requisitos del software

3.1.1. Introducción

En este punto se tratará la Especificación de Requisitos Software o ERS, teniendo como referencia el estándar IEEE 830. Y se abordarán los subpuntos: propósito, ámbito del sistema y visión general del producto.

Propósito

El propósito de las ERS es definir los problemas que se tienen que gestionar. Los problemas son propuestos por el cliente, según sus necesidades y objetivos. Con ello se consigue una mayor facilidad a la hora de implementar y desarrollar el proyecto.

Ámbito del sistema

El trabajo se basa en el desarrollo de un módulo Odoo para gestionar una academia de asignaturas de repaso. Esta aplicación se centrará en el análisis de datos o variables que influyen el centro docente (KPI) para mejorar su rendimiento general y como se trata de una empresa, también su rendimiento económico. Así pues, este módulo dispone con una opción de visualización de gráficos con información destacada. También se podrán introducir datos de los alumnos, profesores, programar clases en un calendario, suscribir a los alumnos y registrar las facturas. Todo esto por la modularidad y herencia de Odoo. Posteriormente, en la plataforma Grafana, se podrán generar paneles con gráficos personalizables para observar y analizar el rendimiento del negocio.

Definiciones

Diferentes términos de importancia se mencionan a lo largo de la memoria, se encuentran definidos en el Glosario. Estos son ERP, definido en el apartado 2.1, MVC, Python, XML y PostgreSQL en el 2.2.3, ETL en el 2.3.1.

Visión general

La visión general pretende dar un enfoque global de los factores y requisitos que influyen sobre el producto. A continuación, enumeramos los subapartados: funcionalidad del sistema,



Desarrollo de un módulo del ERP Odoo para el análisis de datos de una academia

características de los usuarios, restricciones a las que está sujeto, suposiciones, dependencias, requisitos futuros y requisitos específicos. Este último punto, explica las condiciones del sistema para que se cumplan las especificaciones solicitadas.

3.1.2. Descripción general

Perspectiva del producto

El módulo a desarrollar está compuesta de dos softwares que se complementarán, Odoo y Grafana. Será necesario instalar Odoo junto con Python, PostgreSQL y el software de Grafana.

Funcionalidad del sistema

El objetivo de la aplicación es poder analizar datos de interés que ayuden a mejorar la gestión administrativa y económica de una academia de repaso. Para conseguir este fin, se incluyen funcionalidades como la gestión de los estudiantes y los docentes, de las asignaturas y de las facturas, de donde se recopilarán los datos. Las funcionalidades serán:

- Creación, edición y eliminación de alumnos
- Creación, edición y eliminación de profesores
- Gestión de las asignaturas ofrecidas
- Suscripción a las distintas asignaturas
- Gestión de un horario de clases
- Creación de paneles gráficos y su personalización
- Visualización y observación de datos de interés mediante gráficos

Características de los usuarios

Los principales usuarios del sistema serán los administradores, por lo que las funcionalidades estarán centradas principalmente en la gestión de la academia y el análisis de datos. Estas funcionalidades consisten en crear, editar y eliminar contactos de profesores y alumnos, crear, editar y eliminar servicios ofrecidos (asignaturas), gestionar las facturas entrantes y crear, editar y eliminar gráficos de visualización de datos, además de tener acceso total al sistema. Por otro lado estarán los profesores, quienes podrán gestionar el horario de clases y su planificación a través del calendario, así como visualizar datos de sus alumnos.

Restricciones

Se necesitará que el entorno cumpla las siguientes condiciones para su funcionamiento:

- Última versión del Sistema operativo Ubuntu o como mínimo la 20.4.
- Navegador web, actualizado a elección.
- Última versión de PostgreSQL o como mínimo la 12.0.
- Última versión de Python o como mínimo la 3.7 más las dependencias que requiere Odoo.

- Instalación correcta de Odoos 16.
- Instalación de la última versión de Grafana.

Suposiciones y dependencias

Para la puesta en funcionamiento del trabajo se han utilizado las versiones software actualizadas a la última versión, explicadas en el párrafo anterior. No se debería optar por versiones inferiores a las descritas. Su uso o su cambio de versión podrían dar fallos, provocando daño en la aplicación de Odoos o Grafana.

Requisitos futuros

La modularidad de Odoos nos permite ampliaciones como la adición de un módulo para la comunicación entre alumnos y profesores u otro de asignación de tareas a los alumnos.

Requisitos específicos

La siguiente tabla adjunta explicará cómo se mostrarán los requisitos específicos del proyecto, los cuales representan las funcionalidades más utilizadas por los administradores y profesores (**Tabla 2**).

Las tablas cuentan con la siguiente estructura:

Número del requisito	Identificará el requisito con un número.
Nombre del requisito	Identificará el requisito con un nombre.
Descripción	Breve explicación del requisito.
Entrada	Condiciones que se deben cumplir para poder llevar a cabo el requisito.
Proceso	Proceso realizado por el sistema durante la ejecución del requisito.
Salida	Resultado de la ejecución del requisito.

Tabla 2: Modelo

Se dividirán en requisitos que podrán llevar a cabo los administradores, los profesores y los alumnos, teniendo en cuenta la plataforma en la que se llevan a cabo.

Requisitos de los usuarios administradores

Las siguientes tablas muestran los requisitos específicos que el sistema debe cumplir para los usuarios administradores. Estos tienen acceso total al sistema, pudiendo ejecutar todas las funciones, acceder a todos los módulos de ambas plataformas y realizar cambios en ellas (**Tabla 3, Tabla 4, Tabla 5, Tabla 6, Tabla 7, Tabla 8, Tabla 9, Tabla 10, Tabla 11, Tabla 12, Tabla 13, Tabla 14, Tabla 15, Tabla 16, Tabla 17, Tabla 18**).



Desarrollo de un módulo del ERP Odoo para el análisis de datos de una academia

Número del requisito	01
Nombre del requisito	Iniciar sesión administrador Odoo
Descripción	Iniciar sesión en Odoo como administrador para acceder a la aplicación.
Entrada	Requiere tener Odoo instalado e inicializado correctamente. Introducir las credenciales de un usuario administrador previamente registrado en la base de datos.
Proceso	El sistema comprueba las credenciales.
Salida	Si son correctas, se iniciará sesión y se podrá acceder a la aplicación. Si son incorrectas dará error y volverá a pedir las credenciales.

Tabla 3: Iniciar sesión administrador Odoo

Número del requisito	02
Nombre del requisito	Cerrar sesión administrador Odoo
Descripción	El administrador cierra sesión.
Entrada	El administrador clicca en su usuario, cerrar sesión.
Proceso	El sistema cierra la sesión.
Salida	Se redirige al usuario a la página de inicio de sesión.

Tabla 4: Cerrar sesión administrador Odoo

Número del requisito	03
Nombre del requisito	Crear contacto
Descripción	Creación de un contacto de tipo alumno o profesor.
Entrada	Dentro del sistema, entrar en la ventana de contactos, cliccar en nuevo e introducir los datos necesarios para la creación de este, como son nombre, DNI, categoría o email entre otros.
Proceso	El sistema almacena los datos en la base de datos.
Salida	El nuevo contacto queda registrado y se muestra en la ventana de contactos.

Tabla 5: Crear contacto

Número del requisito	04
Nombre del requisito	Eliminar contacto
Descripción	Eliminar un contacto del sistema
Entrada	Dentro del sistema, entrar en la ventana de

	contactos, seleccionar el usuario a eliminar y elegir la acción “Suprimir”.
Proceso	El sistema elimina el contacto y sus datos asociados de la base de datos.
Salida	El contacto eliminado y ya no está registrado ni es accesible.

Tabla 6: Eliminar contacto

Número del requisito	05
Nombre del requisito	Visualizar contacto
Descripción	Acceder al formulario de los datos del contacto.
Entrada	Dentro del sistema, entrar en la ventana de contactos y seleccionar uno en concreto.
Proceso	El sistema obtiene los datos del contacto de la base de datos.
Salida	Se muestra la vista con los datos del contacto.

Tabla 7: Visualizar contacto

Número del requisito	06
Nombre del requisito	Editar contacto
Descripción	Permite editar los datos de un contacto.
Entrada	Dentro del sistema, acceder a la ventana de contactos, visualizar uno y pinchar en “Editar”. Modificar aquellos campos que se deseen sin dejar en blanco los obligatorios y darle a aceptar.
Proceso	El sistema comprueba que los campos obligatorios estén rellenos con información válida.
Salida	En caso de que los datos sean correctos, se verá la vista de datos del contacto actualizada. En caso contrario se informará del error y no se guardarán los cambios.

Tabla 8: Editar contacto

Número del requisito	07
Nombre del requisito	Crear servicio
Descripción	Crear un servicio nuevo para la academia.
Entrada	Dentro del sistema, entrar en la ventana de servicios, clicar en nuevo y rellenar los datos pertinentes.
Proceso	El sistema almacena los datos en la base de datos.
Salida	Los servicios creados son visibles en la ventana de servicios.

Tabla 9: Crear servicio

Número del requisito	08
Nombre del requisito	Eliminar servicio
Descripción	Eliminar un servicio del sistema.
Entrada	Dentro del sistema, entrar en la ventana de servicios, seleccionar el servicio a eliminar y elegir la acción “Suprimir”.
Proceso	El sistema elimina el servicio y sus datos asociados de la base de datos.
Salida	El servicio es eliminado y ya no está registrado ni se puede ofrecer.

Tabla 10: Eliminar servicio

Número del requisito	09
Nombre del requisito	Visualizar servicio
Descripción	Acceder al formulario del servicio.
Entrada	Dentro del sistema, entrar en la ventana de servicios y seleccionar uno en concreto.
Proceso	El sistema obtiene los datos del servicio de la base de datos.
Salida	Se muestra la vista con los datos del contacto.

Tabla 11: Visualizar servicio

Número del requisito	10
Nombre del requisito	Editar servicio
Descripción	Acceder al formulario del servicio y editarlo.
Entrada	Dentro del sistema, entrar en la ventana de servicios, visualizar servicio y pinchar en “Editar”. Modificar aquellos campos que se deseen sin dejar en blanco los obligatorios y darle a aceptar.
Proceso	El sistema comprueba que los campos obligatorios estén rellenos con información válida.
Salida	En caso de que los datos sean correctos, se verá la vista de datos del servicio actualizada. En caso contrario se informará del error y no se guardarán los cambios.

Tabla 12: Visualizar servicio

Número del requisito	11
Nombre del requisito	Facturar servicio
Descripción	Vender un servicio a un contacto y facturarlos.
Entrada	Dentro del sistema, entrar en la ventana Facturas y clicar en el botón “Nuevo”. Seguir un proceso administrativo y gestionar la factura.
Proceso	El sistema sigue el proceso administrativo almacenando los datos en la base de datos según se vayan procesando.
Salida	El servicio queda registrado como presupuesto y pedido de un contacto, con su respectiva factura.

Tabla 13: Facturar servicio

Número del requisito	12
Nombre del requisito	Gestionar eventos en el calendario
Descripción	Gestionar clases en el calendario y añadir/quitar asistentes
Entrada	Acceder a sistema, en la ventana de Calendario y realizar cualquier función de gestión de los eventos del calendario
Proceso	El sistema registra la función realizada sobre el evento y sus características
Salida	La función realizada se muestra en el calendario en la fecha indicada.

Tabla 14: Gestionar eventos en el calendario

Número del requisito	13
Nombre del requisito	Iniciar sesión administrador Grafana
Descripción	Iniciar sesión en Grafana como administrador para acceder a la plataforma.
Entrada	Requiere tener Grafana instalada e inicializada correctamente. Introducir las credenciales de un usuario admin previamente registrado en la base de datos.
Proceso	El sistema comprueba las credenciales.
Salida	Si son correctas, se iniciará sesión y se podrá acceder a la plataforma. Si son incorrectas dará error y volverá a pedir las credenciales.

Tabla 15: Iniciar sesión administrador Grafana

Número del requisito	14
Nombre del requisito	Cerrar sesión administrador Grafana
Descripción	El administrador cierra sesión.
Entrada	El administrador clica en su usuario, cerrar sesión.
Proceso	El sistema cierra la sesión.
Salida	Se redirige al usuario a la página de inicio de sesión.

Tabla 16: Cerrar sesión administrador grafana

Número del requisito	15
Nombre del requisito	Crear Gráfico
Descripción	Creación de un gráfico para la visualización de los datos
Entrada	Dentro de la plataforma Grafana, entrar en Home > Dashboards > New Dashboard e introducir los datos necesarios para su creación, como son la adición de tablas y columnas mediante selección o código sql
Proceso	El sistema genera el gráfico basándose en las instrucciones proporcionadas
Salida	El nuevo gráfico se representa en el panel seleccionado

Tabla 17: Crear gráfico

Número del requisito	16
Nombre del requisito	Visualizar Gráfico
Descripción	Visualización de un gráfico para el análisis de datos
Entrada	Dentro de la plataforma Grafana, entrar en Home > Dashboards y seleccionar el panel a visualizar
Proceso	El sistema aplica los filtros inyectados mediante comandos sql y recupera los datos
Salida	El sistema presenta el panel con los gráficos solicitados

Tabla 18: Visualizar gráfico

Requisitos de los usuarios profesor

Los requisitos específicos de los profesores son los descritos a continuación (Tablas **Tabla 19**, **Tabla 20**, **Tabla 21**, **Tabla 22**). Dado que solo tienen acceso a los módulos de contactos y calendario, sus funcionalidades son restringidas.

Número del requisito	17
Nombre del requisito	Iniciar sesión profesor
Descripción	Iniciar sesión en Odoo como profesor para acceder a la aplicación.
Entrada	Requiere tener Odoo instalado e inicializado correctamente. Introducir las credenciales de un usuario profesor previamente registrado en la base de datos.
Proceso	El sistema comprueba las credenciales.
Salida	Si son correctas, se iniciará sesión y se podrá acceder a la aplicación. Si son incorrectas dará error y volverá a pedir las credenciales.

Tabla 19: Iniciar sesión profesor

Número del requisito	18
Nombre del requisito	Cerrar sesión profesor
Descripción	El profesor cierra sesión.
Entrada	El profesor clic en su usuario, cerrar sesión.
Proceso	El sistema cierra la sesión.
Salida	Se redirige al usuario a la página de inicio de sesión.

Tabla 20: Cerrar sesión profesor

Número del requisito	19
Nombre del requisito	Gestionar eventos en el calendario como profesor
Descripción	Gestionar clases en el calendario y añadir/quitar asistentes como profesor
Entrada	Acceder a sistema, en la ventana de Calendario y realizar cualquier función de gestión de los eventos del calendario
Proceso	El sistema registra la función realizada sobre el evento y sus características
Salida	La función realizada se muestra en el calendario en la fecha indicada.

Tabla 21: Gestionar eventos en el calendario como profesor

Número del requisito	20
Nombre del requisito	Visualizar contacto como profesor
Descripción	Acceder al formulario de los datos del contacto como profesor.
Entrada	Dentro del sistema, entrar en la ventana de contactos y seleccionar uno en concreto.
Proceso	El sistema obtiene los datos del contacto de la base de datos.
Salida	Se muestra la vista con los datos del contacto.

Tabla 22: Visualizar contacto como profesor

3.1.3. Ejemplo de uso concreto

Se desea implementar un módulo para la gestión administrativa y el análisis de datos de una academia de repaso que cuenta con distintos tipos de asignaturas impartidas en diferentes horarios. Se quiere poder gestionar los alumnos y profesores que forman parte de esta, gestionar las asignaturas ofrecidas y el proceso que conlleva la suscripción a estas y su facturación, además de poder analizar estadísticas que permitan tomar decisiones correctas sobre el rumbo de la academia. El uso que se le podría dar es:

1. **Almacenamiento de los alumnos y profesores en la aplicación.** Se crearán contactos para cada uno de ellos con sus datos personales y se indicará si son alumnos o profesores.
2. **Gestión de las asignaturas ofrecidas y su suscripción.** Las asignaturas que quiera impartir la academia serán añadidas al sistema con su respectivo precio y datos adicionales. La suscripción de los alumnos a estas podrá ser llevada a cabo en la aplicación, así como la facturación.
3. **Creación de un horario de clases o eventos.** Se podrán crear clases en el calendario a modo de horario, teniendo en cuenta que alumnos y profesores acudirán a estas y notificar a los alumnos por correo sobre ellas.
4. **Creación de dashboards para para la visualización de los datos.** Se crearán paneles con gráficos a demanda en base a los intereses de la academia.
5. **Visualización de datos de interés.** Se podrán visualizar y analizar datos que reflejen métricas sobre la facturación, los alumnos, las asignaturas y el calendario a petición para facilitar la toma de decisiones.

3.2. Casos de uso

Los casos de uso son diagramas que se utilizan para describir las acciones realizadas por un usuario para llevar a cabo un proceso en un sistema. Estos especifican el comportamiento final del software de una manera gráfica desde el punto de vista del usuario final, en nuestro caso, los administradores de la aplicación y los profesores. Para ello, se va a utilizar el lenguaje estándar de modelaje UML (*Unified Modeling Language*). Se van a dividir en casos de uso de administradores y casos de uso de profesores, teniendo en cuenta la plataforma en el caso de los administradores.

Casos de uso de administradores

Odoo

Se va a comenzar representando el caso de uso genérico que representa todas las acciones a las que tiene acceso el administrador. Empezaremos por los casos de uso en Odoo.

Como podemos ver en la siguiente **Ilustración 2**, puede acceder a los apartados de Calendario, Contactos, Servicios y Facturación, además de a los ajustes técnicos.

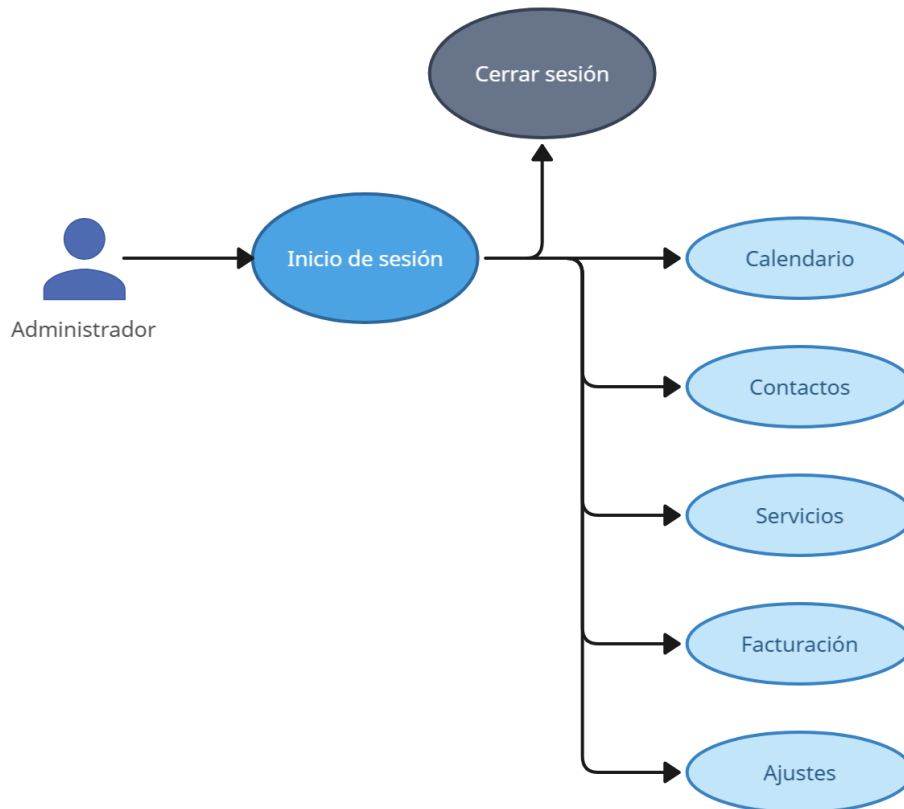


Ilustración 2: Caso de uso general

Pasamos ahora a concretar casos de uso, dentro de estos apartados expuestos. En la Ilustración 3, en el apartado de contactos, podemos acceder a la creación de contactos, y una vez creados, acceder a varias funcionalidades. Estas funcionalidades son: acceso directo a las ventas pertenecientes al contacto, junto a la posibilidad de crear una nueva, acceso directo al calendario del contacto, junto a la posibilidad de asignarle un nuevo evento y acceso directo a la vista de las facturas del contacto. De esta forma, el manejo del módulo es mucho más fluido.

El caso de la Ilustración 4, trata del acceso a los servicios de la academia.

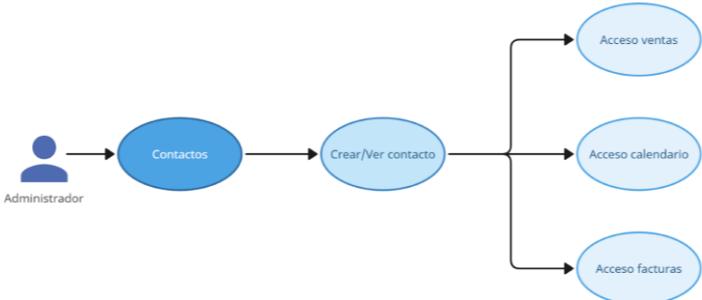


Ilustración 3: Caso de uso contactos

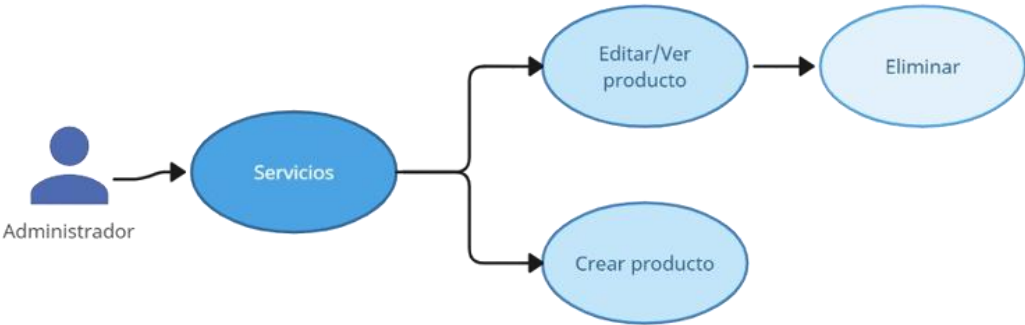


Ilustración 4: Caso de uso servicios

Grafana

En Grafana, se debe poder iniciar sesión, crear Dashboards, personalizar gráficos y añadir base de datos (**Ilustración 5**).

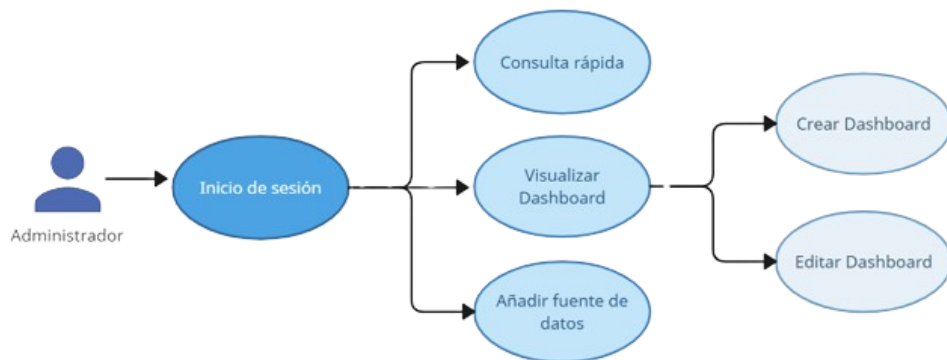


Ilustración 5: Caso de uso Grafana

Casos de uso profesores

Los profesores simplemente van a tener acceso al módulo de contactos y al de calendario. En contactos, van a ser capaces de visualizar los datos de sus alumnos y suyos propios, mientras que en calendario van a poder crear eventos y asignar alumnos para así gestionar sus horarios (**Ilustración 6**).

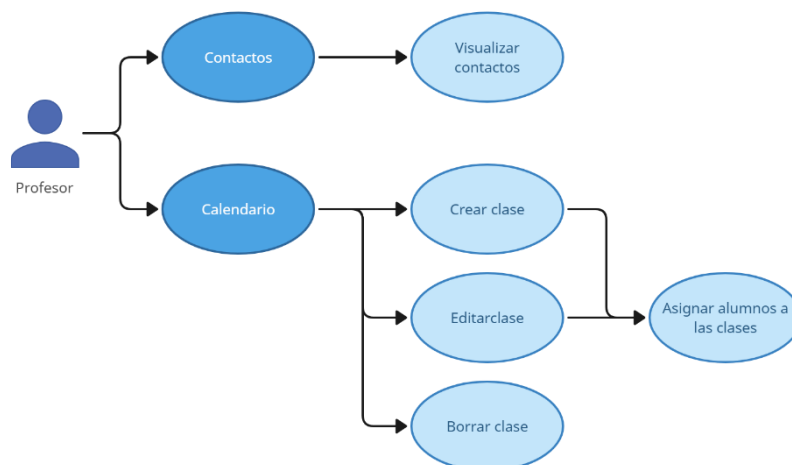


Ilustración 6: Caso de uso profesores

3.3. Identificación y análisis de soluciones posibles

Para la resolución de estos problemas y el cumplimiento de los requisitos expuestos, plantaremos a continuación las soluciones posibles dentro del entorno de trabajo expuesto en los capítulos anteriores.

3.3.1. Odoo

Trabajamos con el ERP Odoo, un software de código abierto y modular. Teniendo en cuenta estas características, podemos valorar las siguientes dos posibilidades.

- Desarrollo de un módulo propio desde cero
- Aprovechar los módulos base para crear nuestra aplicación

La primera de las dos opciones es la que mayor libertad ofrece a la hora de diseñar la aplicación a nuestro gusto. Diseñar un módulo propio permite estructurar y plantear las características deseadas a demanda, creando modelos, vistas y menús originales sin tener que seguir el guion preestablecido por los módulos base. Permite una mayor especialización de la aplicación para el ámbito de mercado en el que se requiere. Los contras, por otra parte, son que esta libertad puede ser abrumadora y llevar a complicar el desarrollo de forma innecesaria. Requiere de mucho conocimiento e inversión de tiempo en el aprendizaje de Odoo, por lo que no es muy recomendable.

La segunda opción es la más viable, ya que se aprovecha toda la estructura ofrecida por Odoo. La aplicación resultante cumplirá con la mayoría de los estándares de negocio, ya que está diseñada para ello. Facilitará mucho su desarrollo, ya que la gran mayoría de las funciones estarán cubiertas. En cambio, esta solución no ofrecerá tanta especialización o no se adaptará plenamente al ámbito de mercado. Las funciones de las que dispone no cubrirán todas las necesidades planteadas, por lo que implicaría tener que adaptarse a los módulos propuestos en vez de adaptar los módulos a nuestras necesidades.

Sin embargo, también es posible desarrollar una solución mixta, la cual consiste en crear un módulo que herede de los módulos base y modificarlos. De esta manera, nos beneficiamos tanto de las funcionalidades ya diseñadas por los especialistas de Odoo como de la libertad de personalización que nos brinda un desarrollo propio. Como es posible combinar la herencia de clases con la creación y expansión de estas, no es necesario dedicarse al desarrollo desde cero, dado que se malgastaría mucho tiempo en desarrollar funcionalidades de las que ya dispone Odoo. Tampoco habría que conformarse con la propuesta base de los módulos, ya que se pueden modificar. Esta flexibilidad es precisamente la que ha llevado a elección de esta solución.

3.3.2. Herramientas de análisis de datos

Las opciones de herramienta de análisis de datos que tenemos disponibles para llevar a cabo nuestro desarrollo han sido expuestas en anteriores secciones. Las herramientas externas son mucho más versátiles y completas que las herramientas que proporciona Odoo. Es por ello que la plataforma elegida para la solución ha sido Grafana. Existen ciertas complicaciones posibles a

la hora de desarrollar una solución conjuntamente con otra plataforma externa, como pueden ser.

- Incompatibilidades a la hora de extraer los datos
- Dificultad añadida de aprender otra plataforma
- Aumento de complejidad de la aplicación final

La incompatibilidad con la base de datos sería la problemática principal a tener en cuenta, sin embargo, las herramientas de análisis de datos de hoy en día están muy preparadas para solucionar estos problemas, siendo Grafana de las mejor preparadas.

La dificultad es subjetiva y depende de la persona que enfrente el problema, pudiendo suponer un inconveniente grave o no suponer ningún problema. La creación de dashboards o paneles se realizará mediante el editor de Grafana en la pestaña Dashboards. A la hora de generar gráficos, disponemos de dos posibilidades. Un editor de sentencias SQL o un editor más gráfico y esquemático llamado Query Builder. Para nuestro ámbito, el editor SQL sería la elección más normal, dado que da una mayor flexibilidad, sin embargo, para un posible usuario menos familiarizado con este lenguaje, el Query Builder podría ser una opción a tener en cuenta.

3.3.3. Disponibilidad de datos

Al tratarse de un desarrollo nuevo, partimos de no tener datos con los que trabajar. No disponemos de alumnos, ni profesores, ni facturas... Dado que nuestra aplicación se basa en el análisis de datos, debemos encontrar una solución para disponer de ellos. Valoraremos las siguientes opciones:

- Extracción, transformación y carga de datos
- Población de datos en Odoos
- Creación manual de datos de prueba

De todas las opciones, la creación manual de datos es la menos atractiva y competente. Primero porque es tediosa y requiere de mucho tiempo para obtener una cantidad de datos de prueba significativa y segundo porque es poco profesional y los resultados a mostrar a la hora de analizar datos no son suficientes.

La extracción, transformación y carga de datos es una opción interesante, ya que Odoos permite la carga de datos demo externos mediante ficheros CSV. Existen herramientas web diseñadas para la generación aleatoria de datos en este formato, por lo que, juntando ambas capacidades, solo queda transformar los datos obtenidos al formato que interpreta Odoos, XML. Tenemos extracción de datos en formato CSV, transformación mediante programación a formato XML y carga de los ficheros resultantes en Odoos en forma de datos demo. Esta opción es válida, sin embargo, no se adapta completamente a los modelos de datos de la plataforma y la transformación se lleva a cabo de forma manual, lo que conllevaría trabajo extra.

Por último, contamos con una solución nativa de Odoos para la problemática. Odoos dispone de un método propio para la población automática de datos mediante programación. Gracias a la línea de comandos llamada Odoos CLI, contamos con los comandos *populate*, los cuales se introducen en la terminal y llaman a una serie de métodos que permiten la generación aleatoria de datos. Mediante esta herramienta se pueden generar una cantidad no trivial de datos específicos del modelo deseado, lo que la convierte en la mejor opción, ya que es la que ofrece mayor flexibilidad y adaptabilidad en el entorno de Odoos. En el capítulo siguiente se detallará más en profundidad esta característica.



4. Diseño de la solución

Una vez analizado el problema de la aplicación que pretendemos desarrollar y expuestos los requisitos necesarios que debe satisfacer e identificar las posibles soluciones, pasamos a diseñar la solución. Para ello vamos a definir primero la arquitectura del sistema a rasgos generales, indicar los módulos de los cuales hará uso la aplicación y explicar su funcionalidad, explicar en profundidad la población de datos mediante Odoo *CLI* y proponer prototipos de la aplicación final.

4.1. Arquitectura del sistema

El primer paso para el diseño de una solución se trata de esquematizar o representar los grandes bloques que la conforman. A continuación, se identificarán los componentes esenciales que forman parte de la solución y se explicará la relación entre ellos (**Ilustración 7**).

- **Usuario.** El sistema necesita un usuario que le dé instrucciones.
- **Computador.** Dispondrá de un sistema operativo Ubuntu 20.4 que contendrá todo el software.
- **Navegador.** Permitirá acceder a las plataformas.
- **Odoo.** ERP que albergará los módulos que componen la aplicación.
- **Grafana.** Herramienta de análisis de datos que permitirá analizar los datos de Odoo.
- **Base de datos.** Tipo PostgreSQL que almacenará los datos provenientes de Odoo y se los proporcionará a Grafana.

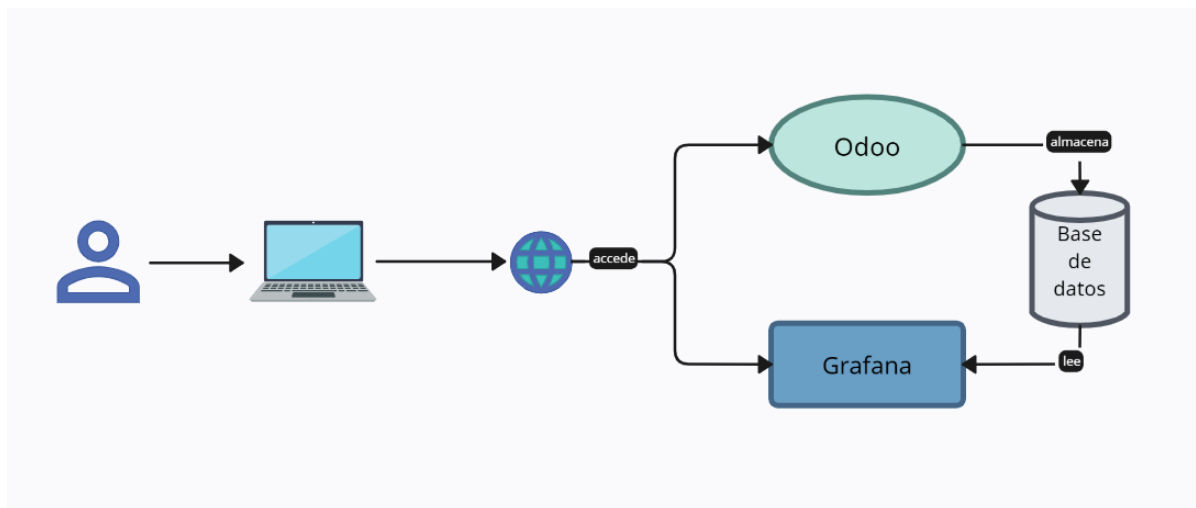


Ilustración 7: Arquitectura del sistema

Esta imagen representa gráficamente los componentes del sistema y su interacción. En primer lugar, el Usuario se conectará al sistema mediante un ordenador con distribución Ubuntu server 20.4 basada en Linux. Este ordenador requerirá de conexión a internet y se conectará, a través de un navegador actualizado, al software ERP Odoo versión 16. Odoo tendrá instalado nuestro

módulo y almacenará nuestros datos en una base de datos PostgreSQL. Posteriormente, la plataforma Grafana leerá esta fuente de datos y mediante consultas SQL representará los gráficos a los cuales accederemos a través del navegador.

4.2. Módulos de Odoo que componen la aplicación

Aquí se van a comentar los módulos que se van a instalar en Odoo. Se comentarán sus funciones base y de qué forma podrán cumplir los requisitos expuestos en el apartado anterior. Se utilizará la herencia para añadir las funcionalidades extra. Los módulos son:

- **Contactos.** El módulo de contactos permite llevar un registro de todos los usuarios y clientes del sistema de forma ordenada y simplificada. Nos permitirá crear, editar y borrar a los alumnos, así como a los profesores, muy cómodamente. Su interfaz se puede visualizar en forma de canvas o de lista, mostrando un formulario cuando se crean o editan los contactos. Se ha heredado para añadir algunos campos de interés que veremos más adelante.
- **Calendario.** Muestra un calendario con la opción de crear eventos programados de forma intuitiva. Permitirá crear clases a una determinada hora de una determinada asignatura, con la opción de asignar alumnos y profesores a ella. Esto podría servir para la planificación de un horario semestral para la gestión de clases. Tanto los administradores como los profesores tienen acceso a este módulo.
- **Ventas.** La funcionalidad del módulo de ventas es la más amplia de todas. Sus principales funciones son el registro de servicios o productos para su compraventa, tanto por parte de los vendedores como de los compradores, manteniendo una lista de estos disponible con sus respectivas características. Nos permitirá crear las asignaturas y ofrecerlas como servicio a nuestros alumnos, pudiéndose acceder a los detalles de estas como su precio o descripción.
- **Facturación.** Contiene los registros de pagos o facturas de los vendedores junto con sus detalles. Es posible ver su estado y seguir su flujo, comprobando si las facturas están pendientes, pagadas, canceladas... Con ello podemos registrar nuestra actividad económica y controlar las cuentas.

4.3. Población de los modelos

La solución elegida para resolver la problemática de datos consiste en la utilización de los comandos que ofrece Odoo *CLI*. Odoo *CLI* es una interfaz de línea de comandos que proporciona una serie de funcionalidades ya implementadas en los modelos de Odoo. Permite acciones como iniciar el servidor, crear un módulo prefabricado o contar las líneas de código totales. En nuestro caso, la acción que nos interesa es la de poblar la base de datos, mediante el comando *populate*.

Odoo populate

Odoo *populate* permite la población de bases de datos mediante su ejecución. La sintaxis consta del nombre del comando, los modelos a poblar y el tamaño de la muestra de datos a generar. Un ejemplo sería: *odoo-bin populate --models res.partner --size small*.



Métodos de population

La clase population alberga los métodos que hacen posible la población de datos, cuatro en total, siendo dos de ellos imprescindibles para su funcionamiento.

- **`_populate_factories()`**: es un método que genera un diccionario de valores de campos y lo devuelve, es decir, un diccionario de tuplas que contienen el nombre del campo y su respectivo valor generado.
- **`_populate(size)`**: es el método que crea los valores que pueblan el modelo en sí. Recibe el parámetro por consola, a través del método `_populate_sizes`.
- **`_populate_sizes`**: devuelve el diccionario que define los tres tamaños posibles ('small', 'medium', 'large') y su número de registros que posteriormente crea `_populate()`.
- **`_populate_dependencies`**: devuelve una lista de modelos que tienen que ser poblados previamente al modelo actual.

Herramientas de población

Existen también las llamadas 'population tools', que son herramientas en forma de métodos que facilitan la generación aleatoria de datos. Son seis en total, pero solamente es necesario conocer sus funciones a grandes rasgos y sus nombres. Son capaces de cosas como devolver un valor aleatorio de una lista, iterar sobre ella o devolver una constante. Estos son *cartesian*, *randomize*, *iterate*, *constant*, *compute* y *randint*.

4.4. Prototipos

El último paso previo a desarrollo de la solución consiste en esbozar mediante un prototipo como quedaría la solución final una vez acabada, para tener un objetivo claro de cómo enfocar el desarrollo (**Ilustración 8**).



Ilustración 8: Pototipo Grafana

Se compondría de gráficos de barras, verticales u horizontales, “pie charts” o gráficos circulares, gráficos temporales y datos concretos. Se pretende que la exposición de los mismos sea variable, que cambie según los intereses de la Academia, para así disponer de la mejor información para la presente toma de decisiones.

5. Desarrollo de la solución

5.1. Introducción al desarrollo

En este capítulo del proyecto se explicará el desarrollo de una manera exhaustiva, incluyendo fragmentos de código que poblarán los modelos de la base de datos y las vistas xml que darán forma a la aplicación y las consultas a la base de datos que se realizarán para mostrar los gráficos. Se describirá la estructura de ficheros que conforma el módulo de Odoo y se explicará que es la herencia y qué importancia tiene en el proyecto y como se han hecho estas consultas SQL para el análisis de datos.

5.2. Directorio del proyecto

Comenzamos por el directorio de la aplicación en Odoo (**Ilustración 9**). La estructura que conforma el módulo está contenida dentro de la carpeta custom y contiene las siguientes carpetas y archivos:

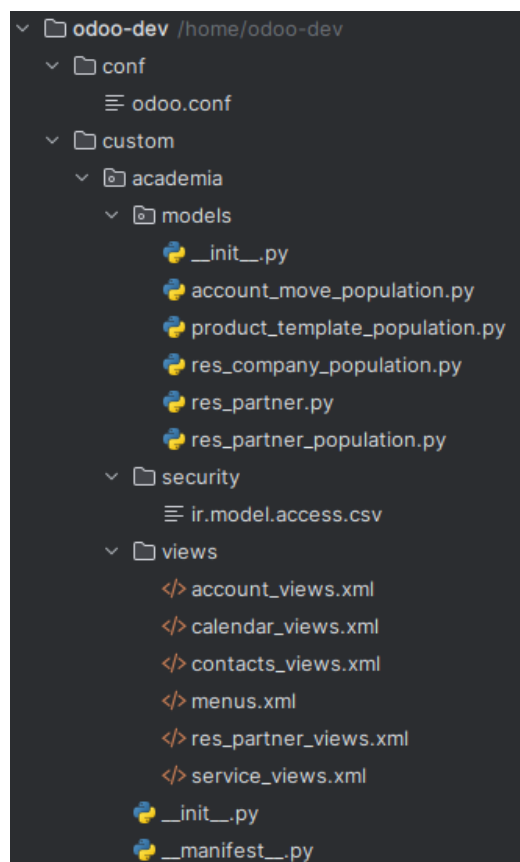


Ilustración 9: Directorio del proyecto

- **Conf.** En este directorio se encuentra el fichero `odoo.conf`, el cual está compuesto por una serie de parámetros que configuran la manera en la que arranca el servidor.
- **Models.** Es el directorio más importante, contiene los archivos python que construyen la parte lógica del módulo. En nuestro caso contiene los archivos python que poblarán la base de datos.
- **Security.** Contiene la configuración de seguridad del proyecto. Básicamente es donde se proporciona acceso a los distintos modelos. Lee archivos en formato csv.
- **Views.** Aquí es donde se encuentran todos los archivos xml que conforman las vistas del módulo.
- **__init__.py.** Es un archivo que contiene la o las carpetas referenciadas de donde extraer los archivos python. Sin la ruta a nuestros archivos `.py`, el sistema no los puede encontrar.
- **__manifest__.py.** Es el archivo principal del módulo, lo equivalente a una carta de presentación. Contiene la información necesaria para que el sistema entienda que se trata de un módulo y una serie de características que lo definen. A continuación se mostrará su estructura y se comentarán sus campos más importantes. Mientras que los campos `name`, `description`, `author` y `version` son autoexplicativos, se destaca el campo `depends`, el cual incluye el nombre de los módulos de los cuales heredará nuestro módulo. Se deben instalar previamente a nuestro módulo, de lo contrario, no funcionará. El campo `data` debe incluir todos los ficheros que siempre deben instalarse con el módulo, es decir, archivos csv o xml que contengan datos, vistas o seguridad (**Ilustración 10**).

```
{
    'name': 'academia',
    'version': '1.0',
    'depends': ['base', 'sale', 'account', 'calendar', 'contacts', 'l10n_es'],
    'author': "Pablo Navalón Arnal",
    'description': """
Módulo para una Academia de repaso.
""",
    'data': [
        'security/ir.model.access.csv',
        'views/menus.xml',
        'views/res_partner_views.xml',
        'views/service_views.xml',
        'views/contacts_views.xml',
        'views/account_views.xml',
        'views/calendar_views.xml',
    ],
    'application': True,
    'installable': True,
}
```

Ilustración 10: Manifest.

5.3. Herencia

La herencia es uno de los mecanismos más utilizados en la programación orientada a objetos. Se basa en la reutilización de código mediante la extensión de clases. Gracias a ella no es necesario reestructurar, rediseñar ni ampliar clases cuyo funcionamiento está testado y libre de errores. Consiste en la creación de una subclase a partir de una clase padre o superclase, de la que obtiene todos sus métodos y atributos. La subclase puede añadir funcionalidades y extender su comportamiento sin romper la jerarquía previa de clases. Así se pueden diseñar clases con funciones más específicas que parten de otras más genéricas.

Entendido el concepto de herencia, pasamos a explicar cómo funciona en Odoo. En Odoo la herencia puede ser utilizada tanto en los modelos (6) como en las vistas (7), aportando fluidez y facilitando la reutilización de código.

5.3.1. Herencia de modelos

Comenzando por los modelos, existen tres tipos de herencia basada en la modularidad:

Herencia clásica

Consiste en la creación de una clase nueva que obtiene todos los campos, métodos y metadatos de una clase existente. Se crea una copia de la clase original y se modifica esta copia dejando la original como estaba. En Odoo esto se consigue haciendo uso de los atributos de modelo `_inherit` y `_name` a la vez. Permitiría a la clase que hereda utilizar los campos y métodos originales, así como sobrescribirlos sin afectar a la clase padre. Se corresponde con la herencia *Prototype*, dentro de *Traditional Inheritance* en la Ilustración 11.

Delegación

Esta herencia es la más flexible de las tres y consiste en una relación donde el modelo en cuestión delega parte de su funcionalidad al modelo original. Es útil para cuando se quiere extender un modelo sin modificarlo directamente, es decir, sin añadir campos o métodos si no que se aprovechan los campos y métodos existentes del modelo original. Esto se consigue mediante los atributos `_name` e `_inherits`, donde `inherits` consiste en una tupla del nombre del modelo heredado y un campo `id`, el cual tendrá una relación *Many2One* con el modelo original. Esto permite que la clase heredada acceda a los campos de la otra clase, pero no a sus métodos. Se corresponde con *Delegation Inheritance* en la Ilustración 11.

Extensión

Se trata del tipo de herencia más utilizada en Odoo. Consiste en crear un nuevo módulo que extienda el modelo original y reemplace su comportamiento. Permite añadir campos, métodos o reconfigurarlo directamente afectando al modelo original. Las posibilidades son:

- Añadir campos al modelo
- Sobrescribir la definición de los campos del modelo
- Añadir restricciones al modelo
- Añadir métodos al modelo
- Sobrescribir los métodos existentes

Se corresponde con *Class Inheritance*, dentro de *Traditional Inheritance*.

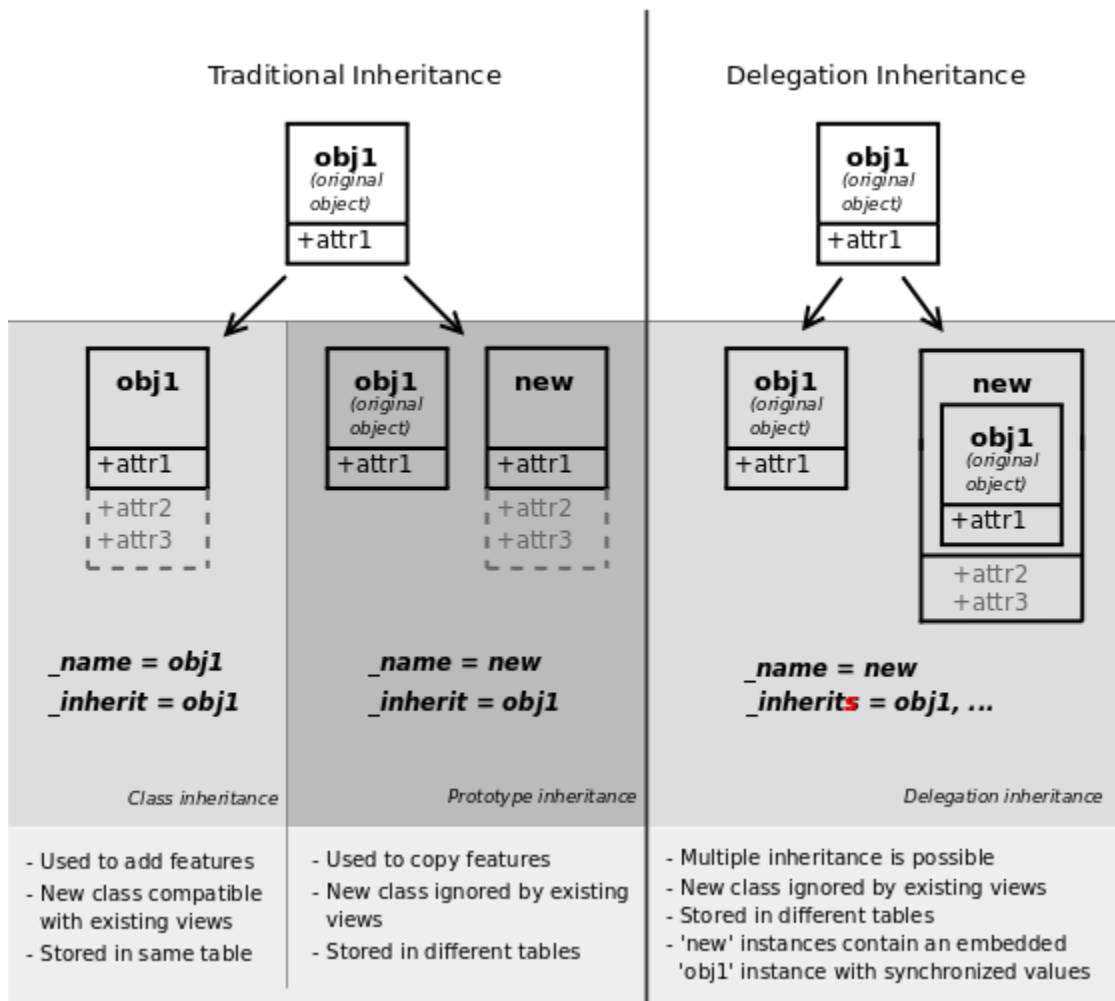


Ilustración 11: Herencia de modelos en Odoo

5.3.2. La herencia de vistas

En Odoo existe la llamada herencia de vistas (**Ilustración 12**), la cual se trata de una herencia que afecta a los ficheros XML que dan forma a la interfaz gráfica de la aplicación. Funciona de manera similar a la herencia de modelos, permitiendo adquirir las estructuras y campos de la vista padre.

Campos de herencia

En el caso de las vistas, esto se consigue especificando los valores de dos campos concretos, *inherit_id*, con su atributo *ref*, y *mode*, el cual puede ser *extension* o *primary*. La especificación del campo *inherit_id* es obligatoria, junto con el atributo *ref* que debe indicar el *id* del modelo padre. El campo *mode*, por lo general solo se cambia a *primary* si la herencia es de tipo delegación; por defecto su valor es *extension*.

```
<field name="inherit_id" ref="library.view_book_form"/>
```

Ilustración 12: Herencia de vistas

Especificaciones de herencia

Estos son elementos cuya función sirve para localizar los elementos a modificar dentro del fichero *xml* de la vista padre. Hay tres formas de localizar los elementos:

- Elemento *xpath* con el atributo *expr*. Se trata de un elemento dentro de la composición de la vista que permite localizar el punto exacto donde realizar la modificación.
- Indicar el nombre del campo a modificar. Es útil cuando la modificación solo afecta al campo en cuestión.
- Indicar el nombre del elemento a modificar. Igual que con el nombre, solo que con cualquier elemento estructural del xml.

Estas especificaciones pueden tener el atributo *position*, que especificará como se modificará el elemento localizado. Por defecto *inside*, pero puede ser *after*, *before*, *attributes* o *move* (Ilustración 13).

```
<xpath expr="page[@name='pg']/group[@name='gp']/field" position="inside">
  <field name="description"/>
</xpath>

<field name="res_id" position="after"/>

<div name="name" position="replace">
  <div name="name2">
    <field name="name2"/>
  </div>
</div>
```

Ilustración 13: Especificaciones de herencia

La herencia obtiene una gran importancia dentro de Odoo, siendo utilizada prácticamente en todos los modelos existentes. En nuestro caso será utilizada tanto en los modelos de población de datos como en las vistas, siendo una parte crucial del desarrollo del proyecto.

5.4. Models

Los modelos en Odoo son las clases que contienen los objetos que se implementan en la aplicación. Están escritos en Python y contienen todos los campos, métodos y restricciones que proporcionan la lógica al módulo. En este caso, nuestro módulo está formado por los siguientes archivos.

El primero de ellos es *__init__.py*, que iniciará el resto de archivos python de la carpeta. Mediante las siguientes líneas de código (Ilustración 14).

```

from . import res_partner
from . import res_partner_population
#from . import res_company_population
from . import product_template_population
from . import account_move_population

```

Ilustración 14: `__init__.py`

El primero de los archivos python se trata de una extensión del funcionamiento de la clase `res_partner`. Se ha decidido ampliar ligeramente la funcionalidad del módulo de contactos para permitir especificar si el contacto se trata de un profesor o de un alumno mediante el modelo `res_partner.py`. Al contar con tantos registros, es útil poder diferenciar entre contactos de profesores y alumnos mediante una búsqueda por un campo específico. El campo `is_student` se trata de un *booleano* que se mostrará en la ficha del contacto. Si es *False*, no mostrará nada, mientras que si está activo, aparecerán los campos **age**, **sex** y **grade**. Siendo el primero de tipo numérico y los dos últimos de selección. Aquí se puede observar la ilustración Ilustración 15.

```

from odoo import models, fields

class ResPartner(models.Model):
    _inherit = 'res.partner'

    is_student = fields.Boolean(string="Alumno", default=True)
    age = fields.Integer(string="Edad")
    sex = fields.Selection(
        [
            ('masculino', 'Masculino'),
            ('femenino', 'Femenino'),
            ('indefinido', 'Indefinido')],
        string='Sexo',
        default='indefinido'
    )
    grade = fields.Selection(
        [
            ('1bach', '1° Bachillerato'),
            ('2bach', '2° Bachillerato')],
        string='Curso',
    )

```

Ilustración 15: `res_partner`

Seguimos a continuación con los archivos python que, heredando de su respectivo modelo original, utilizan los métodos y herramientas de la clase `populate` para poblar tablas. Estos archivos son `res_partner_population.py`, `product_template_population.py` y `account_move_population.py`. Todos los archivos de población se estructuran de la misma manera, cada uno con sus peculiaridades. Es por ello que primero se van a definir los campos y métodos generales de los que todos hacen uso y después se procederá a explicar archivo por archivo sus peculiaridades.

Desarrollo de un módulo del ERP Odoo para el análisis de datos de una academia

Los campos principales de los archivos python de población son:

- **`_inherit`**. Indica el modelo del cual se heredan sus funcionalidades. Se debe cambiar la “_” del nombre del modelo por un “.” para que el sistema lo pueda identificar.
- **`_populate_dependencies`**. Se indican los modelos que se deben poblar previamente al actual.
- **`_populate_sizes`**. Se indica la cantidad de registros a crear, dependiendo del tamaño de la población.

Los métodos comunes a todos los archivos de población son los siguientes:

- **`_populate_factories`**. Contiene las diversas variables y métodos que generarán los valores de los registros.
- **`_populate`**. Creará los registros con los valores de **`_populate_factories`**. Se debe llamar al método **`super._populate()`** para evitar romper el flujo de ejecución.

Comenzamos por el primer modelo para la población de datos, **`res_partner_population.py`**.

Este fichero hereda del modelo **`res_partner`**, el cual es la clase de los contactos, y su método **`_populate_factories`** devuelve valores generados para el nombre, email, teléfono, activo y compañía. Los dos últimos se aportan información sobre si el contacto está en uso o si se trata de una compañía. Además, también generará los valores de edad y curso previamente añadidos con el modelo **`res_partner`**. Las dependencias serán los modelos de las compañías (**`res_company`**) y de las industrias (**`res_partner_industry`**). Los valores se generan mediante los siguientes métodos:

- **`generate_name`**. Combina aleatoriamente valores de entre dos listas para generar un nombre y lo devuelve. Cuenta con una pequeña probabilidad de generar contactos de tipo compañía ya que se requiere para poder continuar con la ejecución del sistema.
- **`generate_is_company`**. Devuelve el booleano resultante del valor de la variable ‘**`is_company`**’.
- **`generate_email`**. Genera un email a partir del nombre del contacto y lo devuelve.
- **`generate_phone`**. Genera un número de teléfono aleatorio y lo devuelve.
- **`generate_is_student`**. Devuelve el booleano resultante del valor de la variable ‘**`is_student`**’.
- **`generate_age`**. Si ‘**`is_student`**’ está a **`True`**, genera una edad de entre una lista con probabilidades y la devuelve.
- **`generate_grade`**. Funciona igual que **`generate_age`** pero con una lista de cursos.

Estos métodos se crean con la ayuda de las herramientas de población mencionadas en el apartado 4.x. En este caso, se utiliza ***compute*** para estos cuatro métodos y ***constant*** para que el valor de ***active*** sea siempre ***True***. Para finalizar, dentro de **`_populate`** se llama a método ***super***, como ya hemos comentado previamente.

El siguiente modelo se trata de **`product_template_population.py`**. Hereda del modelo **`product_template`**, el cual es la clase de los productos, y su método **`_populate_factories`** devuelve valores generados para el nombre, el tipo, la categoría y el precio. La variable categoría necesita ser extraída del modelo **`product_category`**, por lo que se añade su dependencia en la variable **`_populate_dependencies`** y de paso se añade la herencia y

personalización de las categorías en nuestro modelo también. Así podemos generar las categorías a nuestro gusto, dentro del mismo archivo que `product_template`. Esto se consigue heredando del modelo `product_category`. Su método `_populate_factories`, devuelve solamente el nombre de la categoría mediante la herramienta `compute` y el siguiente método:

- `generate_category_names`. Itera sobre una lista de nombres de categorías de productos y devuelve una por ejecución hasta que salta la excepción.

Volviendo a la generación de las variables de `product_template`, disponemos de los siguientes métodos.

- `generate_unique_name`. Itera sobre una lista de nombres de producto y devuelve un nombre por ejecución hasta que salta la excepción.
- `generate_category_id`. Recupera el valor temporal devuelto por `generate_unique_name` y lo busca en una serie de listas. Dependiendo de la lista en la que lo encuentre, asigna una categoría u otra a una variable y la devuelve.

Como se puede observar, estos métodos no generan categorías ni productos combinando valores de manera aleatoria, sino que generan una cantidad fija de registros con nombres concretos. Esto se ha programado de esta manera porque no es realista que una academia de repaso disponga de una cantidad excesiva de productos o servicios que no respetan la unicidad, pudiendo darse el caso de ofrecer cincuenta asignaturas donde cinco serían matemáticas, tres lenguas, etc. Esta problemática se habría resuelto de forma más sencilla creando los valores a mano, pero por respetar la automaticidad del proceso de población, se ha adaptado su creación a los métodos de Odoopopulate.

A continuación, viene el modelo `account_move_population.py`, el cual hereda del modelo `account_move`. Este modelo forma parte de las facturas en Odoop y su generación es la más compleja de las tres. Su método `_populate_factories` devuelve valores generados para el cliente, el tipo de factura, su estado, su fecha y sus líneas de compra con los productos seleccionados. Para que las facturas sean coherentes, este modelo toma los nombres de los clientes y productos generados en los dos modelos anteriores, por lo que sus dependencias especificadas en `_populate_dependencies` son `res_partner` y `product_template`. Los métodos que generan los valores son los siguientes:

- `generate_partner_id`. Devuelve un id de partner aleatorio de entre la lista previamente poblada del modelo `res_partner`, siempre y cuando el campo `is_student` del propio partner esté a `True`.
- `generate_invoice_date` y `generate_invoice_due_date`. Devuelven una fecha aleatoria comprendida entre hoy y hace un año.
- `generate_invoice_line`. Genera una línea de factura eligiendo un producto aleatoriamente de entre la lista previamente poblada del modelo `product_template`. La línea de factura se compone del nombre, id, precio y cantidad del producto.

Las herramientas de población utilizadas son `compute`, para los métodos de generación aleatoria comentados, y `constant` para los valores de tipo de factura y estado, que deben ser siempre los mismos para que el sistema permita generarlos. Finalmente se llama al método `super` de `_populate`.

Acabados los archivos de población, se ha decidido ampliar ligeramente la funcionalidad del módulo de contactos para permitir especificar si el contacto se trata de un profesor o de un alumno mediante `res_partner.py`. Al contar con tantos registros, es útil poder diferenciar entre



contactos de profesores y alumnos mediante una búsqueda por un campo específico. El campo **is_student** se trata de un *booleano* que se mostrará en la ficha del contacto. Si es *False*, no mostrará nada, mientras que, si está activo, aparecerán los campos **age**, **sex** y **grade**. Siendo el primero de tipo numérico y los dos últimos de selección.

Terminamos aquí con la población de modelos.

5.5. Views

Las views o vistas en Odoo son los archivos escritos en código XML que dan forma a la interfaz visual de Odoo. El sistema después los transforma a lenguaje HTML para ser representados en la web.

Hemos comentado previamente que nuestro desarrollo aprovecha las funcionalidades de los módulos de ventas, contactos, calendario, y facturación. Sin embargo, estos módulos son en sí mismo una aplicación propia, por lo tanto, para unificarlos y poder acceder a ellos desde una misma interfaz, se ha creado un menú común desde donde acceder directamente a las propias vistas de cada uno de los módulos. La vista XML que se mostrará en la pantalla principal será **menus.xml** (Ilustración 16).

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

  <menuitem id="academia_root_menu"
            name="Academia"
            sequence="0"
          />

  <menuitem id="service_menu"
            name="Servicios"
            parent="academia_root_menu"
            sequence="30"
          />

  <menuitem id="contacts_menu"
            name="Contactos"
            parent="academia_root_menu"
            sequence="20"
          />

  <menuitem id="account_menu"
            name="Facturas"
            parent="academia_root_menu"
            sequence="40"
          />

  <menuitem id="calendar_menu"
            name="Calendario"
            parent="academia_root_menu"
            sequence="15"
          />

</odoo>
```

Ilustración 16: Menus.xml

Esta vista se compone de los llamados **menúitem**, que se convertirán en los accesos directos a las vistas propias de cada módulo, visibles en la barra superior de la pantalla principal del módulo academia. Los **menúitem** se componen de los atributos **id**, **name**, **parent** y **sequence**. **Parent** especifica el menú del cual cuelgan y **sequence** permite personalizar su orden de aparición en la barra.

Las vistas heredadas son **res_partner_views.xml**, **calendar_views.xml**, **contacts_views.xml**, **account_views.xml** y **service_views.xml**. De entre ellas, solo **res_partner_views.xml** es una vista personalizada, añadiéndole los campos creados mediante el modelo **res_partner.py** a la vista original del modelo **res_partner**. Como se aprecia en la imagen expuesta a continuación, a la vista resultante se le atribuye un nuevo **id** y se le indica su **model**, “ir.ui.view”, que quiere decir que es una vista. Los siguientes campos le atribuyen su nombre, el modelo que representará y la vista que heredará y reemplazará. Esto se consigue mediante una **referencia** al id de la vista original en el campo **inherit_id**. Después, se programa el atributo **xpath** para buscar un campo con **expr**, el cual contendrá un campo de la vista original, y una posición en la que debe introducirse el nuevo campo mediante **position**. El campo **es_alumno** se insertará detrás de un campo llamado “vat”, mientras que el resto solo serán visibles siempre y cuando **es_alumno** sea True (**Ilustración 17**).

```
<?xml version="1.0" encoding="utf-8"?>
<odoo>
  <record id="view_partner_form_inherit" model="ir.ui.view">
    <field name="name">res.partner.form.inherit</field>
    <field name="model">res.partner</field>
    <field name="inherit_id" ref="base.view_partner_form"/>
    <field name="arch" type="xml">
      <xpath expr="//field[@name='vat']" position="after">
        <field name="es_alumno"/>
      </xpath>

      <xpath expr="//field[@name='vat']" position="after">
        <field name="edad" attrs="{ 'invisible': [('es_alumno', '=', False)] }"/>
        <field name="sexo" attrs="{ 'invisible': [('es_alumno', '=', False)] }"/>
        <field name="curso" attrs="{ 'invisible': [('es_alumno', '=', False)] }"/>
      </xpath>
    </field>
  </record>
</odoo>
```

Ilustración 17: Res_partner_views.xml

El resto de las vistas simplemente cuelgan de su menú padre, el cual a su vez, cuelga del menú principal de la academia. Solo falta recalcar que mediante el atributo **action** se llama a la acción que muestra la vista del módulo original (**Ilustración 18**).

```
<?xml version="1.0" encoding="utf-8"?>
<odoo>
  <menuitem id="contacts_submenu"
    name="Alumnos"
    parent="contacts_menu"
    action="contacts.action_contacts"
    sequence="112"
  />
</odoo>
```

Ilustración 18: Contacts_views.xml

Acabada la parte de Odoo, comenzaremos con el desarrollo de los dashboard en Grafana.



5.6. Consultas SQL

Grafana permite la creación de dashboards e inserción de gráficos para el análisis de datos. Para que se permita la creación de un gráfico, solo es necesaria una conexión a una fuente de datos y la realización de una consulta sql. Estas consultas extraen información concreta de una base de datos y la modelan para que pueda representarse de forma legible y útil en gráficos de datos. Para nuestro proyecto, dada la limitada disponibilidad de datos, el dashboard diseñado se podría considerar una demo, ofreciendo en realidad una grandísima variedad de opciones de análisis.

En nuestra plataforma Grafana, se ha creado un dashboard principal llamado Academia, que albergará todos los gráficos diseñados para el análisis de datos de nuestra academia. Existen distintos tipos de gráficos, algunos de los más interesantes se explicarán a continuación:

- **Barras.** Representa un conjunto de datos mediante barras rectangulares de forma proporcional a su valor.
- **Circular.** Representa conjuntos de datos de forma proporcional en un gráfico con forma de pastel.
- **Gauge.** Es un medidor radial que muestra la medida de un valor en contraste con unos valores absolutos o porcentuales.
- **Histograma.** Es una variante del gráfico de barras pero utilizada para mostrar la distribución de una sola variable.
- **Barras de gauge.** Una variante del gráfico gauge que muestra contraste con valores absolutos o porcentuales en forma de barras.

Los gráficos cuentan con características personalizables, muchas de las cuales son comunes a todos, mientras que otras son propias de cada tipo. De entre las personalizaciones comunes, se pueden resaltar:

- **Opciones de panel.** Aquí se configura el título y la descripción del gráfico.
- **Leyenda.** Permite configurar donde se ve, qué valores muestra y si es visible o no.
- **Límites.** Permite configurar límites en forma de valores absolutos o porcentuales. Si los valores los cruzan, se colorean a elección.
- **Asignación de valores.** Sobrecribir valores de la gráfica a posteriori de su ejecución.

Nosotros contamos con datos generados provenientes de unas pocas tablas, comentadas en el apartado anterior, por lo que nuestras consultas se han reducido a extraer información sobre los alumnos, asignaturas y beneficios económicos. Las consultas siguen todas una estructura similar, la cual consiste en:

- **SELECT.** Seleccionamos los campos que queremos que nos muestre la gráfica, con la posibilidad de renombrarlos con *AS* o agregarlos mediante *COUNT*, *DISTINCT* o *SUM*, entre otros.
- **FROM.** Se indican las tablas que se van a consultar para extraer los datos. Estas serán **res_partner**, **product_product**, **product_template**, **product_category**, **account_move** y **account_move_line**. Contamos con la posibilidad de realizar consultas a varias tablas a la vez mediante *JOIN*, para lo que será necesario darle nombre a la tabla para poder referenciarla.
- **WHERE.** Permite filtrar las búsquedas en base a las condiciones que se indiquen.
- **GROUP BY.** Aquí se indican los campos que se quieren agrupar para mostrar el resultado por categorías de valores.
- **ORDER BY.** Ordena los resultados de la búsqueda de manera ascendente *ASC* o descendente *DESC*.

A continuación, se va a detallar cada una de las consultas realizadas y comentar el resultado obtenido.

Categorías de asignaturas más populares

Se trata de un gráfico **circular** que muestra visualmente qué categorías de asignaturas son las más populares en base a su porcentaje de ventas. Para ello seleccionamos los nombres de las categorías y los id de producto agregados mediante *COUNT*. Indicamos las tablas de las cuales extraemos los datos y las unimos mediante *JOIN*. Filtramos solo por las facturas publicadas y de tipo venta y finalmente agrupamos por nombre de categoría y ordenamos de manera descendente. Con esto conseguimos obtener las veces que una asignatura de una determinada categoría ha aparecido en una línea de factura. La consulta es la siguiente **Ilustración 19**.

```
SELECT
    pc.name AS categoria,
    COUNT(pt.id) AS total_asignaturas
FROM
    account_move_line aml
JOIN
    account_move am ON aml.move_id = am.id
JOIN
    product_product pp ON aml.product_id = pp.id
JOIN
    product_template pt ON pp.product_tmpl_id = pt.id
```

```
JOIN
|   product_category pc ON pt.categ_id = pc.id
WHERE
|   am.state = 'posted'
|   AND am.move_type = 'out_invoice'
GROUP BY
|   pc.name
ORDER BY
|   total_asignaturas DESC;
```

Ilustración 19: Categorías de asignaturas más populares

Con el resultado obtenido la academia puede valorar en qué tipo de asignaturas son las que los alumnos necesitan más apoyo y, por ejemplo, reforzar la docencia u ofertar más plazas.

Ventas totales por asignatura

En este caso tenemos un gráfico de barras gauge que nos muestra la cantidad total de asignaturas vendidas. Los valores a representar son los nombres de las asignaturas y su cantidad. Para ello en la consulta seleccionamos los nombres de las asignaturas y su cantidad en una suma agregada mediante *SUM*. Se indican las tablas correspondientes y se enlazan mediante sus ids. Por último, agrupamos por nombre de asignatura y ordenamos de forma descendente según su cantidad vendida. Con esto conseguimos obtener la suma de todas las veces que una asignatura aparece en una línea de factura (**Ilustración 20**).

```
SELECT
|   pt.name AS "Asignatura",
|   SUM(aml.quantity) AS "Cantidad Vendida"
FROM
|   account_move_line aml
|   JOIN product_product pp ON aml.product_id = pp.id
|   JOIN product_template pt ON pp.product_tmpl_id = pt.id
GROUP BY
|   pt.name
ORDER BY
|   "Cantidad Vendida" DESC
```

Ilustración 20: Ventas totales por asignatura

Este gráfico permite a la academia identificar qué asignatura históricamente ha sido la más vendida, pudiendo así ofertar más plazas o aumentando la docencia.

Ganancias anuales

Este gráfico es de tipo **series temporales** y muestra las ganancias obtenidas durante el último año. Los valores que queremos representar son las ganancias en el eje Y y los meses en el X. Seleccionamos las fechas de las facturas truncadas a meses y la suma de su precio total. Se filtra solo por facturas de venta que estén publicadas y se agrupa y ordena por mes. Con esto conseguimos obtener una gráfica que nos permite visualizar las ganancias del último año agrupadas por la suma de las ventas totales de cada mes (**Ilustración 21**).

```
SELECT
  DATE_TRUNC('month', invoice_date) AS Mes,
  SUM(amount_total) AS Dinero_mensual
FROM
  account_move
WHERE
  state = 'posted'
  AND move_type = 'out_invoice'
GROUP BY
  DATE_TRUNC('month', invoice_date)
ORDER BY
  DATE_TRUNC('month', invoice_date) ASC;
```

Ilustración 21: Ganancia anuales

Esta información ayuda a la academia a conocer su situación financiera a lo largo de los meses durante el último año, por lo que podría tomar decisiones económicas con más conocimiento

Objetivo de ganancias mensuales

Se trata de un gráfico de tipo **gauge** que muestra las ganancias obtenidas durante el mes actual. Se seleccionará el mes actual y la suma agregada de sus facturas de la tabla **account_move**, filtrando únicamente las facturas de venta publicadas. Se agrupa por los meses de las fechas de factura. El resultado nos muestra la suma de las ganancias de este mes, comparada con las ganancias de los meses anteriores, mediante un porcentaje. El código resultaría como en la **Ilustración 22**.

```
SELECT
  DATE_TRUNC('month', invoice_date) AS month,
  SUM(amount_total)
FROM
  account_move
where
  state = 'posted'
  AND move_type = 'out_invoice'
group by
  DATE_TRUNC('month', invoice_date)
```

Ilustración 22: Ganancias mensuales

La academia podrá identificar si los beneficios económicos obtenidos este mes son suficientes comparados con los beneficios de meses anteriores.

Alumnos mensuales

Se trata de dos gráficos de tipo **gauge** en un mismo panel que muestran la evolución mensual del número de alumnos, uno del mes actual y otro del previo. Se debe seleccionar la fecha de las facturas por meses y la cuenta única agregada con *DISTINCT* de los ids de los alumnos que han facturado este mes, obteniendo los datos de las tablas **account_move** y **res_partner**. Se filtran las facturas para que solo sean de ventas, que estén publicadas y que se extraigan solamente facturas que se hayan publicado durante el último mes. En el caso del segundo gráfico, es lo mismo pero el filtro extrae solamente facturas del mes previo al actual. Por último, se agrupa por mes. Lo que nos muestran estos dos gráficos es la comparativa de alumnos apuntados a nuestra academia en el mes actual frente al previo (**Ilustración 23**).

```
SELECT
  DATE_TRUNC('month', am.invoice_date) AS month,
  COUNT(DISTINCT rp.id) AS total_alumnos
FROM
  account_move am
  JOIN res_partner rp ON am.partner_id = rp.id
WHERE
  am.state = 'posted'
  AND am.move_type = 'out_invoice'
  AND EXTRACT(MONTH FROM am.invoice_date) = EXTRACT(MONTH FROM CURRENT_DATE)
GROUP BY
  DATE_TRUNC('month', am.invoice_date)
SELECT
  DATE_TRUNC('month', am.invoice_date) AS month,
  COUNT(DISTINCT rp.id) AS total_alumnos
FROM
  account_move am
  JOIN res_partner rp ON am.partner_id = rp.id
WHERE
  am.state = 'posted'
  AND am.move_type = 'out_invoice'
  AND EXTRACT(MONTH FROM am.invoice_date) = EXTRACT(MONTH FROM CURRENT_DATE - INTERVAL '1 month')
GROUP BY
  DATE_TRUNC('month', am.invoice_date)
```

Ilustración 23: Alumnos mensuales

La información permitiría a la academia darse cuenta de si están atrayendo alumnos o perdiéndolos y tomar decisiones en cuanto a su gestión.

Número de alumnos por edades

Es un histograma que muestra la cantidad de alumnos para cada edad, así como su media. Seleccionamos la edad y los cursos de la tabla **res_partner**, filtramos que tenga el campo **is_student** a **True** y que la edad y el curso no estén vacíos. Con esto ya tenemos un gráfico que nos muestra la distribución de las edades de nuestros alumnos (**Ilustración 24**).

```

SELECT
  grade as curso,
  age
FROM
  res_partner
WHERE
  is_student = True
AND grade <> ''

```

Ilustración 24: Número de alumnos por edades

Permite a la academia adaptarse mejor a las necesidades de sus clientes al conocer más concretamente características como su edad.

El resultado final de la adición de todos los gráficos al dashboard permite a la academia obtener a primera vista una buena cantidad de información y métricas sobre el rendimiento del negocio. Les permite tomar decisiones basadas en información real y actualizada de forma que puedan tomar decisiones que mejoren el rumbo del negocio. El resultado es el siguiente: **Ilustración 25**.

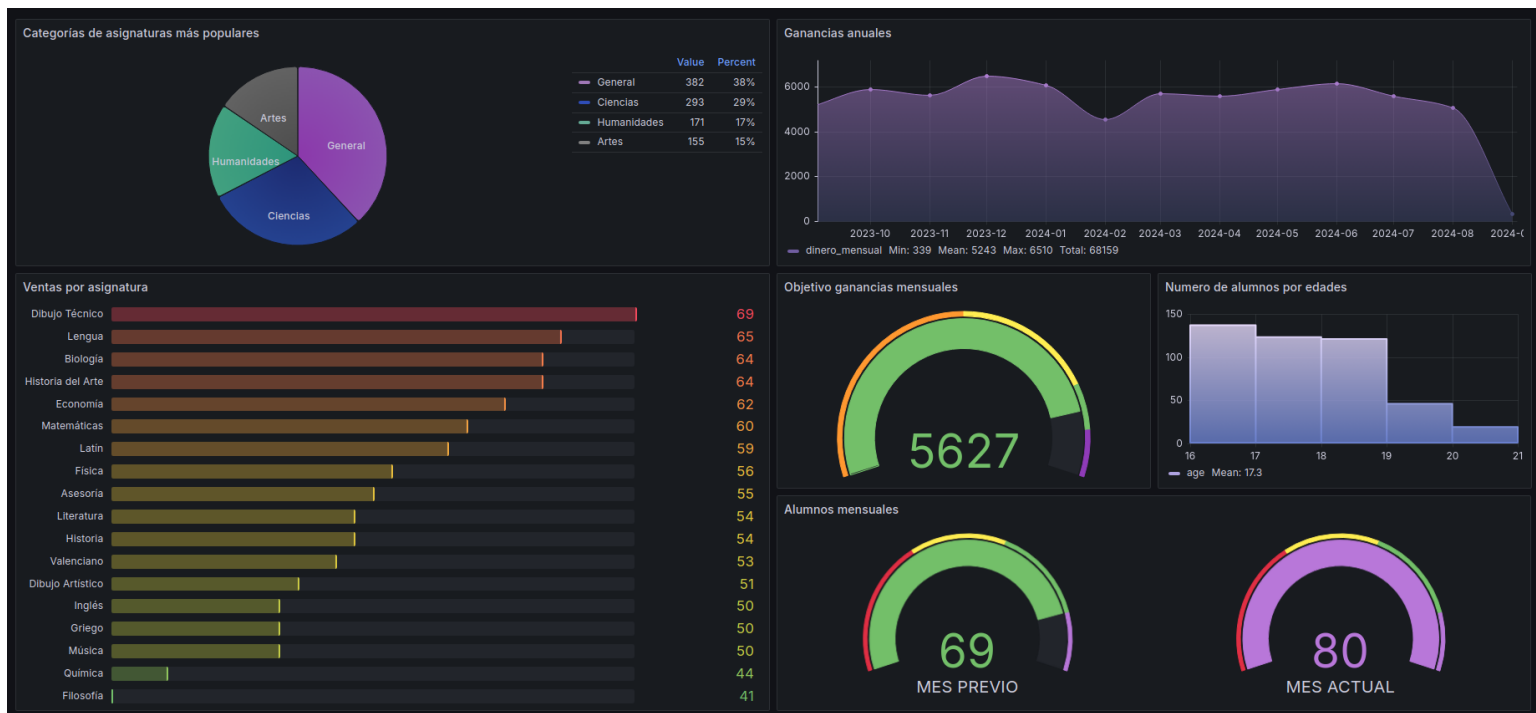


Ilustración 25: Dashboard academia



6. Implantación

6.1. Configuración de la máquina virtual y las plataformas Odoo y Grafana

Para la configuración del entorno se ha de utilizar una máquina virtual con sistema operativo Linux, concretamente Ubuntu 20.04. Para ello, en primer lugar, se debe descargar el software que permite la ejecución de máquinas virtuales, como por ejemplo VirtualBox o VMware Player. En este caso se ha utilizado VirtualBox, el cual se puede descargar en <https://www.virtualbox.org/wiki/Downloads>.

El siguiente paso es descargar la imagen ISO de la máquina virtual. Como ya se ha mencionado, esta será Ubuntu 20.04 server, disponible para descargar en <https://ubuntu.com/download/server>.

Una vez hecho esto, se creará una máquina virtual de 20GB de almacenamiento dinámico y 3072MB de RAM y se procederá con la instalación y la configuración. Se han seguido los pasos del tutorial que ofrece la propia página de Ubuntu <https://ubuntu.com/tutorials/install-ubuntu-server#1-overview>.

Una vez terminada la configuración, se instalará una interfaz gráfica para facilitar la implementación del software.

```
Sudo apt install gnome-shell ubuntu-gnome-desktop -y
```

En caso de querer acceder al servidor de forma remota, podemos hacer uso del servicio SSH, u otros como RPD.

6.1.1. Odoo

Ahora procedemos a instalar Odoo en nuestro sistema. Disponemos de varias opciones, instalación mediante **paquetes**, instalación del código directamente desde el **repositorio** o instalación por medio de **Docker**. En el caso de los desarrolladores es más conveniente obtener el código mediante la segunda opción, haciendo uso de git, ya que permite una mayor flexibilidad a la hora de trabajar con Odoo.

Primero de todo, se actualizarán e instalarán algunos paquetes requeridos por el sistema, así como git para poder obtener el código del repositorio.

```
Sudo apt-get update  
sudo apt-get upgrade  
sudo apt-get install git
```


A continuación, clonamos Odoo en nuestro sistema desde el repositorio de GitHub, ya sea mediante https o ssh, no sin antes haber creado una carpeta para almacenar todo lo relacionado con Odoo.

```
mkdir home/odoo-dev
cd home/odoo-dev
git clone https://github.com/odoo/odoo.git -b 16.0 --depth=1
```

Seguimos con la instalación de Python 3 y pip.

```
sudo apt-get install python3-dev python3-pip
```

Odoo hace uso de las bases de datos de PostgreSQL, por tanto es necesaria su instalación así como la creación de un usuario.

```
sudo apt-get install postgresql
sudo createuser --superuser $(whoami) o sudo su -c "createuser -s $(whoami) "
postgres
createdb $USER
psql $DB_NAME (para conectarse a la base de datos)
```

Quedan las dependencias de Odoo. Estas pueden ser instaladas mediante paquetes específicos para Ubuntu/Debian o mediante pip.

```
cd /odoo-dev/odoo
sed -n -e '/^Depends:/,/^Pre/ s/ python3-\.*/python3-1/p' debian/control | sudo xargs
apt-get install -y
```

Como últimos detalles, instalamos npm, nodejs, css y wkhtmltopdf.

```
sudo apt-get install npm
sudo ln -s /usr/bin/nodejs /usr/bin/node
sudo npm install -g less less-plugin-clean-css
```

Ahora ya podemos crear un directorio nuevo *custom* con la carpeta *academia*, donde crearemos los archivos necesarios para el funcionamiento de nuestro módulo. Para que Odoo reconozca que hay un **addon**, es necesario que existan los archivos `__init__.py` y `__manifest__.py` como mínimo.



```
mkdir -p ~/odoo-dev/custom/academia
touch ~/odoo-dev/custom/academia/__manifest__.py
touch ~/odoo-dev/custom/academia/__init__.py
```

Para iniciar el servicio de Odoo lo hacemos con el comando desde el directorio de Odoo:

```
./odoo/odoo-bin --addons-path=custom,odoo/addons
```

Aunque es preferible crear un archivo conf e iniciar el servicio mediante este. Para conectarse a Odoo es tan simple como abrir el navegador y conectarse a localhost a través del puerto 8069.

```
http://localhost:8069
```

6.1.2. Odoo CLI

Nuestro módulo cuenta con modelos para la población de datos, por lo que será necesario ejecutarlos previo al acceso de Grafana, pues sin datos en la base de datos, las consultas no funcionarán. Para ello es necesario disponer previamente del módulo Academia instalado. Esto se puede conseguir de dos formas, desde la pestaña de aplicaciones en Odoo o mediante la consola (**Ilustración Ilustración 26**). El comando que pone en marcha su instalación es el siguiente, donde en `-d` se especifica la base de datos y en `-i` el módulo a instalar:

```
/home/odoo-dev$ ./odoo/odoo-bin --addons-path=odoo/addons,custom -d academial -i academia
09:51:48,814 70892 INFO ? odoo: Odoo version 16.0
09:51:48,815 70892 INFO ? odoo: Using configuration file at /home/pablo/.odoorc
09:51:48,815 70892 INFO ? odoo: addons paths: ['/home/odoo-dev/odoo/odoo/addons', '/home/pabl
09:51:48,815 70892 INFO ? odoo: database: default@default:default
```

Ilustración 26: Intalación del módulo

Con la consiguiente finalización de la instalación (**Ilustración Ilustración 27**):

```
INFO academial odoo.modules.loading: Module spreadsheet_dashboard_sale loaded in 0.48s, 32 q
INFO academial odoo.modules.loading: 57 modules loaded in 80.38s, 30272 queries (+30272 extr
INFO academial odoo.modules.loading: Modules loaded.
INFO academial odoo.modules.registry: Registry loaded in 83.537s
```

Ilustración 27: Fin instalación

Una vez instalado el módulo, podemos ejecutar el comando `populate` en la consola desde el directorio de Odoo. `Populate` inicia la población de datos en la base de datos

indicada en `-d`, en los modelos especificados en `-models` y con una muestra de datos del tamaño indicado en `-size`, que puede ser *small*, *medium* o *large*. (Ilustración 28)

```
/home/odoo-dev$ ./odoo/odoo-bin --addons-path=odoo/addons,custom populate -d academial --models account.move --size large
```

Ilustración 28: Comando para la población de la base de datos

Dando como resultado la población del módulo junto con sus dependencias (Ilustración 29):

```
INFO academial odoo.addons.academia.models.res_partner_population: Generating is_student
INFO academial odoo.addons.academia.models.res_partner_population: True
INFO academial odoo.addons.base.populate.res_partner: Setting companies
INFO academial odoo.addons.phone_validation.tools.phone_validation: The 'phonenumbers' Python module is not installed, contact numbers
INFO academial odoo.cli.populate: Populated database for model res.partner (total: 7.851551s) (average: 15.671758ms per record)
INFO academial odoo.cli.populate: Populating database for model product.category
INFO academial odoo.addons.product.populate.product: Set parent/child relation of product categories
INFO academial odoo.cli.populate: Populating database for model product.template
INFO academial odoo.addons.product.populate.product_template: Set barcode on product variants (15)
INFO academial odoo.cli.populate: Populating database for model account.move
INFO academial odoo.models: Batch: 1000/1000
INFO academial odoo.addons.account.populate.account_move: Posting Journal Entries
INFO academial odoo.cli.populate: Populated database for model account.move (total: 150.657479s) (average: 150.506972ms per record)
```

Ilustración 29: Resultado de la población

Ahora ya disponemos de todo para pasar a la plataforma Grafana.

6.1.3. Grafana

A continuación, se instalará Grafana en el sistema. La versión elegida es la 11.1.4, de la Edición OSS. Se hará mediante su paquete deb, a través de los siguientes comandos para Ubuntu o Debian.

```
sudo apt-get install -y adduser libfontconfig1 musl
wget https://dl.grafana.com/oss/release/grafana_11.1.4_amd64.deb
sudo dpkg -i grafana-enterprise_11.1.4_amd64.deb
```

Para inicializar Grafana.

```
sudo /bin/systemctl start grafana-server
```

Por último para conectarnos a Grafana se utiliza también el navegador a través de localhost, puerto 3000.

```
http://localhost:3000
```

7. Pruebas

Una vez se ha analizado el problema, diseñado, desarrollado e implantado la solución, el siguiente paso consiste en realizar las pruebas de funcionamiento para comprobar que el desarrollo funciona como se espera. Para estas pruebas se van a tener en cuenta el cumplimiento de los requisitos específicos expuestos en el apartado de análisis del problema **3.1.2**.

Se va a realizar una demostración del funcionamiento del sistema pasando por todas las funcionalidades disponibles, mencionando qué requisito se está cumpliendo en cada momento al hacerlo. Comenzamos con el inicio de sesión en Odoo (**Ilustración 30**) y la visualización de la primera pantalla que aparece nada más acceder al módulo (**Ilustración 31**), donde se puede observar los accesos directos del menú principal en la barra superior. Se cumplen los requisitos **Tabla 3** y **Tabla 19**.

Correo electrónico

Contraseña

Iniciar sesión

[¿No tienes una cuenta?](#) [Restablecer contraseña](#)

Ilustración 30: Iniciar sesión Odoo

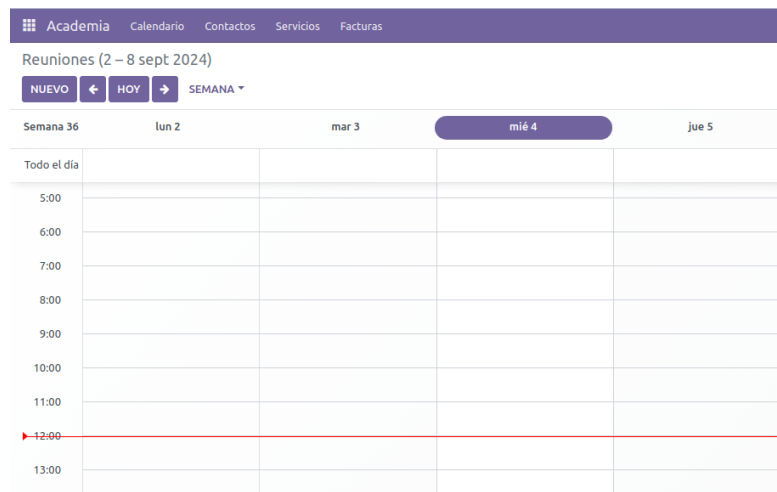


Ilustración 31: Vista inicial

Lo primero que realizaremos será el acceso al menú de contactos (**Ilustración 32**). Una vez dentro, aparecerá una pantalla y podremos crear un contacto pulsando en el botón *Nuevo* de arriba a la izquierda de la pantalla (**Ilustración 33**). Una vez dentro del formulario del contacto, se dará la posibilidad de rellenar todos los campos que se deseen, además de indicar si el contacto es alumno o no, haciendo visibles los campos de edad, sexo y curso si se marca la

casilla (**Ilustración 34**). Para guardar los datos introducidos se debe pulsar en el icono de la nube de esa misma imagen. Si por el contrario, se desea borrar el contacto, dentro del propio formulario se pincha en el botón de Acción arriba a la derecha y se suprime (**Ilustración 35**). Se cumplen los requisitos **3, 4, 5, 6 y 20**.



Ilustración 32: Acceso menú contactos

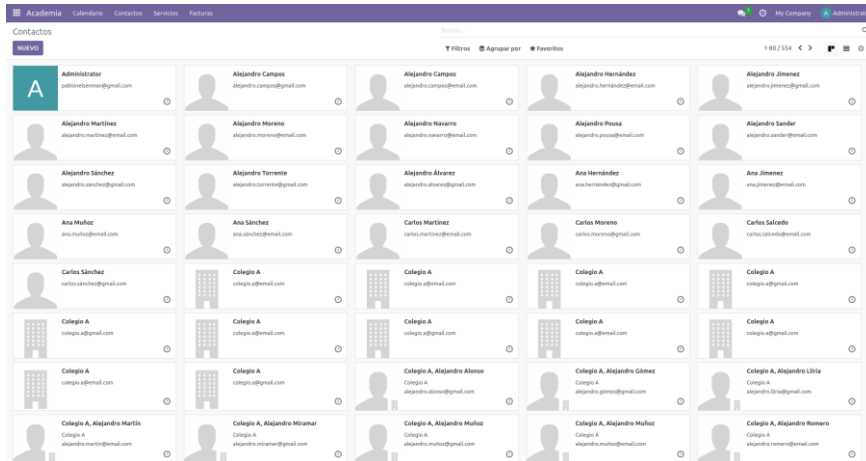


Ilustración 33: Menú contactos

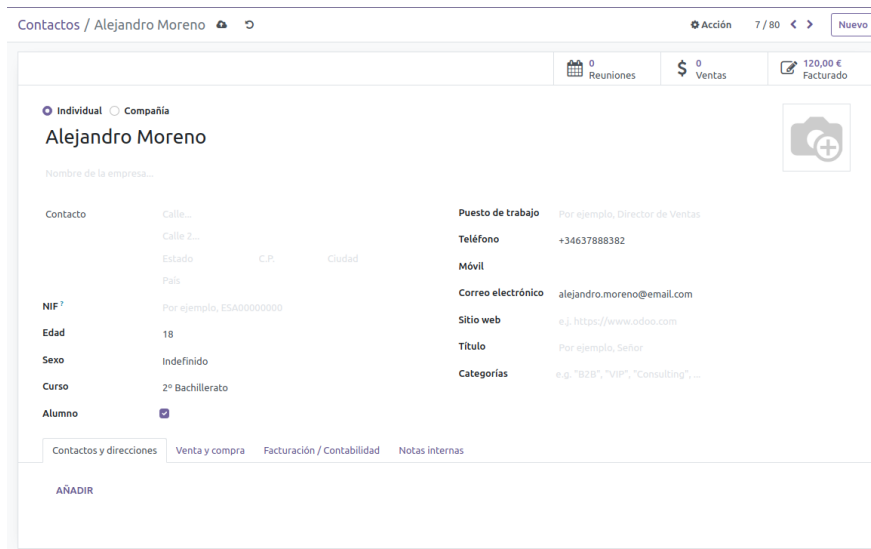


Ilustración 34: Formulario contacto

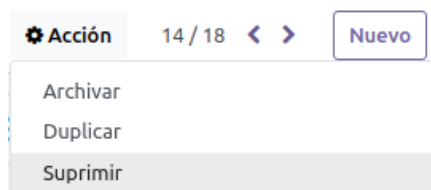


Ilustración 35: Suprimir

Desarrollo de un módulo del ERP Odoo para el análisis de datos de una academia

Seguimos con el acceso a los servicios que ofrecerá la academia. Para visualizar la vista de los servicios, se pulsará en el menú Servicios -> Asignaturas (**Ilustración 36**), resultando en (**Ilustración 37**). Para crear un servicio nuevo, en nuestro caso asignatura, se pulsará en el botón Nuevo, idéntico al de la vista de contactos. Para editar uno existente, se pincha en el elegido. Nos aparecerá un formulario con los posibles campos a rellenar y/o editar (**Ilustración 38**) y basta con pulsar en el icono de la nube para guardarlo. Su eliminación se lleva a cabo de la misma manera que con los contactos (**Ilustración 35**). Quedan cubiertos los requisitos 7, 8, 9, 10.



Ilustración 36: Acceso menu servicios

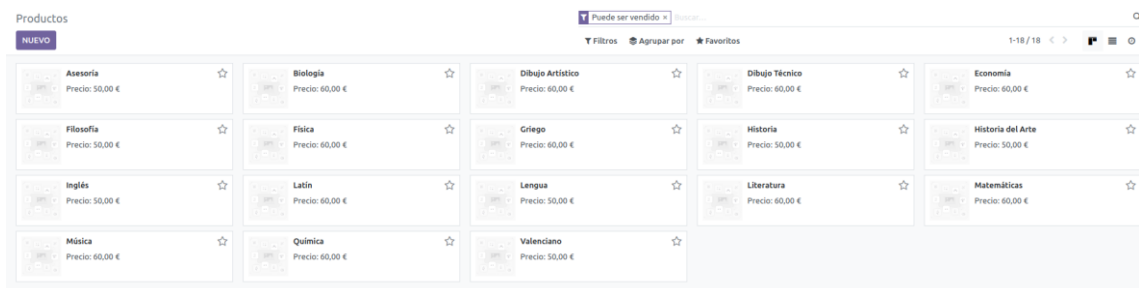
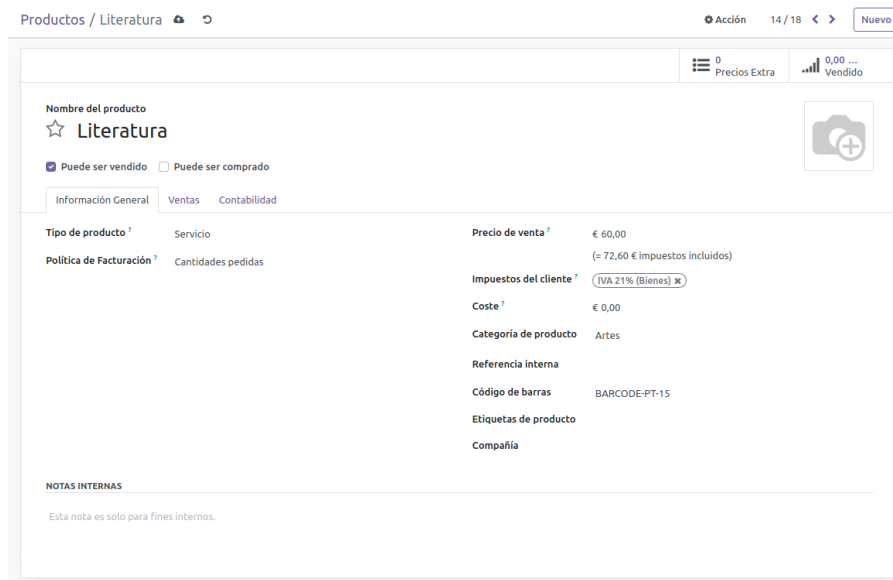


Ilustración 37: Menú servicios



Nombre del producto	
☆ Literatura	
<input checked="" type="checkbox"/> Puede ser vendido <input type="checkbox"/> Puede ser comprado	
Información General Ventas Contabilidad	
Tipo de producto [?]	Servicio
Política de Facturación [?]	Cantidades pedidas
Precio de venta [?]	€ 60,00 (= 72,60 € impuestos incluidos)
Impuestos del cliente [?]	(IVA 21% (Bienes) x)
Coste [?]	€ 0,00
Categoría de producto	Artes
Referencia interna	
Código de barras	BARCODE-PT-15
Etiquetas de producto	
Compañía	
NOTAS INTERNAS	
Esta nota es solo para fines internos.	

Ilustración 38: Formulario servicios

El acceso a la vista de facturas es como los anteriores, a través del menú superior Facturas -> Facturas por clientes (**Ilustración 39**), lo que nos lleva a la lista principal de facturas (**Ilustración 40**), donde habrá que pulsar nuevamente en el botón de Nuevo. Se nos abre el formulario de creación de facturas, se rellena convenientemente (**Ilustración 41**) y se pulsa en el botón de confirmar. Por último, para cubrir la gestión de eventos con el calendario, se pulsa en su botón Calendario -> calendario clases (**Ilustración 42**) y se permite la creación de un evento pulsando en cualquier lugar de la tabla del calendario, apareciendo así el formulario de creación (**Ilustración 43**). Se cubren los casos 11 , 12 y 19.

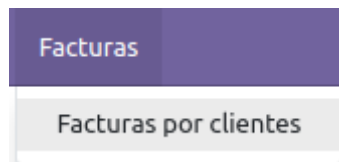


Ilustración 39: Acceso menu facturas

Número	Cliente	Fecha de factura	Fecha de vencimiento	Actividades	Base imponible	Total	Total en divisa	Estado de pago	Estado
INV/2024/00077	Alejandro Campos	28/01/2024	28/01/2024	○	50,00 €	60,50 €	60,50 €	No pagado	Publicado
INV/2023/00014	Alejandro Campos	07/09/2023	07/09/2023	○	60,00 €	72,60 €	72,60 €	No pagado	Publicado
INV/2024/00532	Alejandro Campos	14/07/2024	52 días tarde	○	60,00 €	72,60 €	72,60 €	No pagado	Publicado
INV/2023/00331	Alejandro Campos	28/12/2023	28/12/2023	○	60,00 €	72,60 €	72,60 €	No pagado	Publicado
INV/2023/00179	Alejandro Hernández	05/11/2023	05/11/2023	○	60,00 €	72,60 €	72,60 €	No pagado	Publicado
INV/2023/000163	Alejandro Martínez	31/10/2023	31/10/2023	○	60,00 €	60,50 €	60,50 €	No pagado	Publicado
INV/2024/00384	Alejandro Martínez	24/05/2024	24/05/2024	○	60,00 €	72,60 €	72,60 €	No pagado	Publicado
INV/2023/00016	Alejandro Martínez	08/09/2023	08/09/2023	○	50,00 €	60,50 €	60,50 €	No pagado	Publicado
INV/2024/00239	Alejandro Martínez	31/03/2024	31/03/2024	○	60,00 €	72,60 €	72,60 €	No pagado	Publicado
INV/2024/00045	Alejandro Martínez	19/01/2024	19/01/2024	○	60,00 €	72,60 €	72,60 €	No pagado	Publicado
INV/2024/00347	Alejandro Moreno	11/05/2024	11/05/2024	○	60,00 €	72,60 €	72,60 €	No pagado	Publicado
INV/2024/00098	Alejandro Moreno	03/02/2024	03/02/2024	○	60,00 €	72,60 €	72,60 €	No pagado	Publicado
INV/2023/00335	Alejandro Navarro	30/12/2023	30/12/2023	○	60,00 €	72,60 €	72,60 €	No pagado	Publicado
INV/2023/00166	Alejandro Navarro	01/11/2023	01/11/2023	○	50,00 €	60,50 €	60,50 €	No pagado	Publicado
INV/2024/00260	Alejandro Navarro	06/04/2024	06/04/2024	○	50,00 €	60,50 €	60,50 €	No pagado	Publicado

Ilustración 40: Menú facturas

Facturas / Nuevo

CONFIRMAR VISTA PREVIA BORRADOR PUBLICADO

Factura de cliente Borrador

Cliente Pedro Torrente

Fecha de factura 04/09/2024

Referencia de pago

Fecha de vencimiento 18/09/2024

Moneda EUR

Producto	Etiqueta	Cantidad	Precio Impuestos	Subtotal
Economía	Economía	1,00	60,00 (IVA 21% (Bienes))	60,00 €

Importe libre de impuestos: 60,00 €
IVA 21%: 12,60 €
Total: 72,60 €

Ilustración 41: Formulario facturas

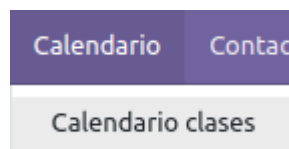


Ilustración 42: Acceso menu calendario

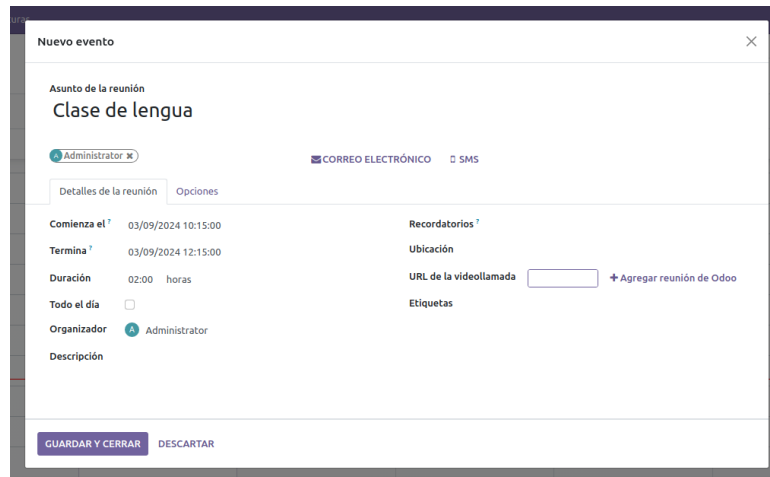


Ilustración 43: Formulario calendario

Las funcionalidades del módulo de la academia quedan cubiertas, solo falta el requisito de cerrar sesión **2** y **18**, que se realiza pinchando en el icono de nuestro usuario arriba a la derecha y seleccionando cerrar sesión en la selección (**Ilustración 44**). Con esto termina el paso por Odoo.

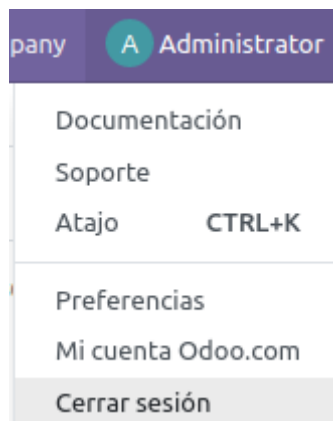


Ilustración 44: Cerrar sesión Odoo

Para los requisitos **13**, **15** y **16** de Grafana, primero iniciamos sesión a través de la pantalla de inicio de sesión en la web (**Ilustración 45**). Una vez dentro, pulsamos en el icono principal de la parte izquierda con las tres rayitas, situado al lado de Home, donde se nos desplegarán los nombres de los menús disponibles (**Ilustración 46**). Nos interesa el menú Dashboards (**Ilustración 47**). Nos aparecerá una lista con los dashboards de los que disponemos y podremos pinchar en el que queramos visualizar. Para crear uno nuevo simplemente tendremos que pinchar en el botón de Nuevo a la derecha del todo. Ya para terminar, se cerrará la sesión **14** desde el icono de usuario arriba a la derecha, al estilo de Odoo (**Ilustración 48**).

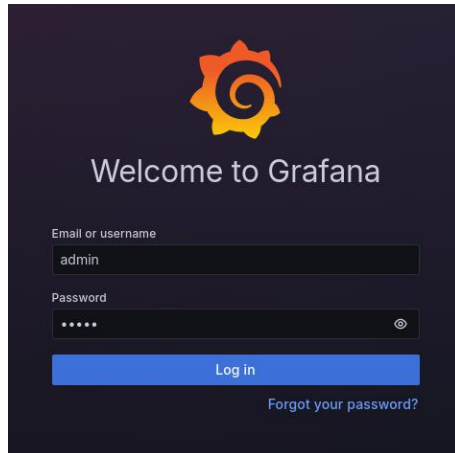


Ilustración 45: Inicio sesión Grafana

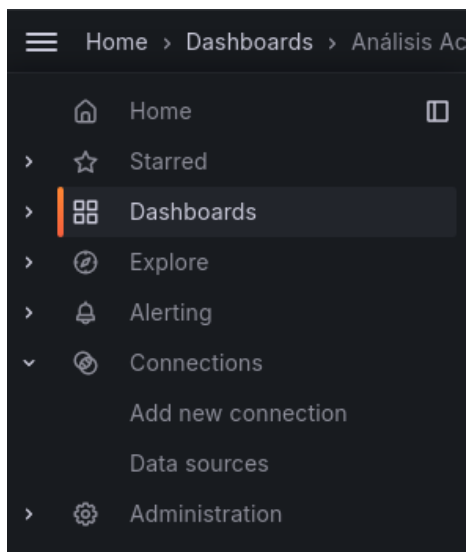


Ilustración 46: Menú inicial

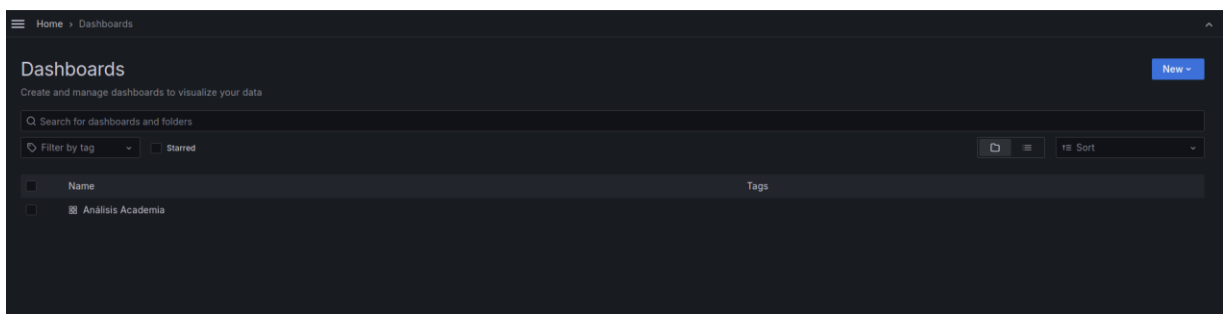


Ilustración 47: Menú dashboards

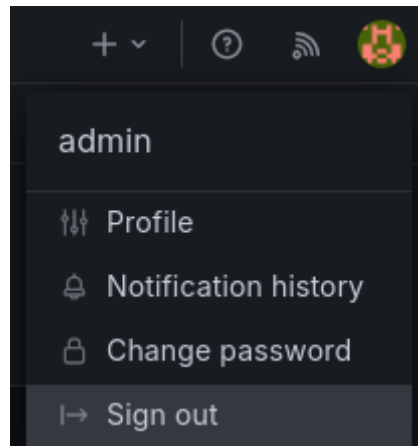


Ilustración 48: Cerrar sesión Grafana

8. Conclusiones

Cumplimiento de los objetivos

Poniendo en retrospectiva los objetivos planteados al principio de la memoria 1, podemos identificar si se han cumplido o no. Comenzando por los objetivos sobre el aprendizaje de los sistemas ERP y la profundización sobre Odoo, se han cumplido con creces puesto que la memoria contiene una gran cantidad de información sobre el qué, el cómo y el cuándo de los ERP. También se han podido resolver todas las problemáticas surgidas durante el desarrollo técnico, como han sido la creación de nuevos módulos, modelos y vistas. La instalación de las plataformas y la posterior planificación del desarrollo ha sido un trabajo un poco más costoso, ya que un desarrollo de esta duración puede resultar abrumador al principio, ralentizando los primeros pasos, sin embargo, una vez puesto en marcha el desarrollo, todo se ha vuelto mucho más fluido.

Problemas encontrados durante el desarrollo

Aprendizaje de Odoo y su ORM. Odoo está construido con los lenguajes Python y XML. Durante la carrera se ha utilizado Python, aunque no demasiado, suficiente para comprenderlo bien y no encontrar prácticamente dificultades a la hora de trabajar con él. Sin embargo, Odoo utiliza su llamada ORM API para programar sus módulos de una manera concreta, por lo que esto sí que ha supuesto una dificultad añadida en la que gastar horas de aprendizaje. El lenguaje XML también ha sido necesario aprenderlo desde cero, aunque no tiene gran complejidad.

Odoo, el análisis de datos y la población de modelos. A la hora de diseñar la solución, me he topado con que Odoo en su versión libre dispone de muy pocas herramientas para el diseño de gráficos que me permitiesen mostrar datos y analizarlos. También me encontré con que no disponía de ninguna fuente de datos de la que extraer, por lo que tenía que contar con datos demo. Se tuvieron que analizar diversas opciones externas para el análisis de datos, así como valorar la manera de inyectar datos en Odoo. La decisión tomada fue la de utilizar Grafana y poblar la base de datos mediante las herramientas CLI que ofrecía Odoo, lo que implicó aprender sobre herramientas de análisis de datos y sobre las herramientas de Odoo.

Herramientas de análisis de datos. Durante el grado se ha llegado a utilizar herramientas como PowerBI o RapidMiner, pero no se había profundizado demasiado en la representación de los datos. Ha hecho falta investigar sobre diversas herramientas y plantear pros y contras sobre cuál debería ser la opción a elegir. Para las consultas a la base de datos era necesario saber SQL, cosa que no ha supuesto apenas dificultad porque ha sido un lenguaje bastante utilizado en la carrera.

Escritura de la memoria en Word. Siempre se menosprecia la dificultad que supone escribir una memoria o un documento de Word y uno no sabe los dolores de cabeza que pueda dar hasta que se enfrenta a ello. La inserción de referencias, los estilos, la armonía de los párrafos y los títulos también han supuesto una dificultad añadida.



Trabajos futuros

Expansión de las funcionalidades del módulo academia. Las funciones disponibles para la academia a día de hoy están limitadas. Se cubren las funcionalidades básicas que requeriría cualquier negocio, adaptadas a una academia de repaso. Actualmente se centran en proveer un calendario para las clases, llevar un listado de alumnos y profesores, ofertar servicios en forma de asignaturas y gestionar las facturas. Hay una infinidad de posibles expansiones que facilitarían la gestión y ampliarían la capacidad de análisis de datos en Grafana. Por hoy el software no permite el acceso a los estudiantes, esta funcionalidad se podría añadir por medio de la adición de un control de asistencias, lo que proporcionaría datos nuevos para su análisis como el ratio de asistencia por clase.

Se podría incluir el sistema de pagos dentro del sistema, ya que por ahora simplemente se registran como pagados sin ofrecer ningún comprobante. Se podría añadir un formulario para la adición de las notas de los alumnos, un espacio para mandar tareas, un chat directo entre alumnos y profesores...

En definitiva, Odoo ofrece mucha libertad en cuanto a personalización e sus módulos. Su desarrollo requiere una cantidad considerable de tiempo y esfuerzo, pero con dedicación, se pueden crear muchas mejoras.

Expansión de los dashboard de Grafana. La adición de nuevos paneles gráficos con nuevas consultas que brinden más información vendría ligada de la mano con la extensión de las capacidades del módulo en Odoo. Cuantas más tablas y campos, más posibilidades de análisis de datos.

Relación con los estudios cursados y aprendizajes

Para concluir la conclusión, la relación con los estudios cursados. La rama que cursé es la de sistemas de información, familiarizándome con la gestión de las tecnologías de la información, el comportamiento organizativo, los sistemas de las organizaciones y las bases de datos, entre otros conocimientos. También entramos en contacto con los ERP, mediante la realización de prácticas con SAP y con el análisis de datos, o más concretamente los procesos ETL mediante herramientas como PoweBI. Este proyecto profundiza sobre los ERP, permitiéndome informarme sobre sus usos, su historia, sus objetivos, etc y me ha otorgado una visión más amplia de la materia. Por otro lado, el análisis de datos ha sido un nuevo entorno de conocimiento en el cual he tenido que profundizar, mientras que con los entornos de las bases de datos y sus lenguajes ya estaba más familiarizado. En definitiva, he podido profundizar más sobre estas temáticas y hacerme una idea más concreta sobre sus aplicaciones reales.

Bibliografía

1. *Motivación, en qué ayuda un ERP*. Recuperado el 7 de marzo de 2024, de <https://nextech.pe/la-importancia-de-contar-con-un-erp/>
2. *Historia del ERP*. Recuperado el 13 de marzo de 2024, de <https://www.velneo.com/blog/historia-de-erp-pasado-presente-y-futuro>
3. *Evaluando ERP - Características de los ERP y sus tipos*. Recuperado el 15 de marzo de 2024, de <https://www.evaluandoerp.com>
4. *Principales ERP*. Recuperado el 18 de marzo de 2024, de <https://www.softwaretestinghelp.com/best-erp-software-systems>
5. *Extract, Transform and Load*. Recuperado el 12 de abril de 2024, de https://es.wikipedia.org/wiki/Extract._transform_and_load
6. *Odoo Inheritance*. Recuperado el 20 de junio de 2024, de <https://www.odoo.com/documentation/15.0/developer/reference/backend/orm.html#inheritance-and-extension>
7. *Odoo Views Inheritance*. Recuperado el 21 de junio de 2024, de <https://www.odoo.com/documentation/16.0/developer/reference/backend/views.html#inheritance>
8. *The Odoo Story*. Obtenido de https://www.odoo.com/es_ES/blog/odoo-news-5/the-odoo-story-56
9. *Odoo Wikipedia*. Obtenido de <https://es.wikipedia.org/wiki/Odoo>
10. *Architecture overview*. Obtenido de https://www.odoo.com/documentation/16.0/developer/tutorials/getting_started/01_architecture.html
11. *Grafana Docs*. Obtenido de <https://grafana.com/docs/grafana/latest/>
12. *Metabase*. Obtenido de <https://www.metabase.com/>
13. *Superset*. Obtenido de <https://superset.apache.org/>

Anexos - Glosario

ERP: Enterprise Resource Planning

ERS: Especificación de Requisitos Software

BOM: Bill of Materials **IMC:** Inventory Management and Control

MRP: Manufacturing Resource Planing

SaaS: Software as a Service

SAP: Systems, Applications, Products in Data Processing

CRM: Customer Relationship Management

LGPL: Lesser General Public License

HTML: HyperText Markup Language

XML: Extensible Markup Language

CSV: Comma-Separated Values

ORM: Object Relational Mapping

CLI: Command-Line Interface

W3C: World Wide Web Consortium

MVC: Modelo Vista Controlador

UML: Unified Modeling Language

ETL: Extract, Transform and Load

SQL: Structured Query Language

API: Application Programming Interface

CSS: Cascading Style Sheets

ISO: Archivo informático que almacena una imagen exacta de un sistema de archivos.

Dashboard: Panel que alberga distintos gráficos para la representación de datos.

Anexos – Objetivos de desarrollo sostenible (ODS)

OBJETIVOS DE DESARROLLO SOSTENIBLE

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

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

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

Este proyecto se centra en el desarrollo de un módulo en Odoo para el análisis de datos de una academia de repaso, por lo que se centra en el área de la educación y el ámbito laboral. De los 17 ODS existentes, es posible relacionarlo con la educación de calidad, el trabajo decente y crecimiento económico, la industria, innovación e infraestructuras y en menor medida con la producción y el consumo responsables.

En primer lugar, el ODS más relacionada se trata de la número 4, ofrecer educación de calidad. Este proyecto, al basarse en la optimización de la gestión y el análisis de datos, puede mejorar la calidad educativa gracias a una mejora en la toma de decisiones. Permite una mejor optimización de los recursos ya que es más fácil distribuirlos si se tiene la información necesaria. Como por ejemplo utilizar las aulas de mayor tamaño para las aulas con más afluencia o distribuir el profesorado en base a la cantidad de alumnos. Facilita el progreso de los estudiantes y mejora su personalización del aprendizaje, gracias a disponer de una tecnología para almacenar datos de forma más eficiente que podrían ser de ayuda para los estudiantes. En general contribuye a disponer de una educación de refuerzo más organizada y efectiva.

En segundo lugar, se relaciona con el ODS número 8 trabajo decente y crecimiento económico. Un desarrollo de este tipo facilita la gestión de una academia, lo cual hace que una entidad educativa funcione de manera más eficiente. Al funcionar más eficientemente, se traduce en una mejora del crecimiento económico, lo que puede brindar más oportunidades de negocio al tener la oportunidad de contratar a más profesores y personal administrativo de la educación, al ofrecer más plazas de una determinada asignatura en base a su demanda, la compra concreta de materiales (ya sea informáticos o prácticos) necesarios para las aulas en base a su consumo.

Se relaciona también con el ODS número 9, industria, innovación e infraestructura. La adición de tecnologías ERP, el análisis de datos y las plataformas Odoo y Grafana suponen una innovación que puede mejorar la infraestructura educativa. Esto es un vivo ejemplo de cómo el uso de herramientas tecnológicas para gestionar academias de repaso contribuye a la modernización del sector educativo, facilita el acceso a la información y mejora su manejo.

Por último, se puede relacionar con el ODS número 12, producción y consumo responsables, aunque con un impacto menos notable que los anteriores. Los ERP son software que se basan en la optimización de los recursos. Aplicado al ámbito escolar, es cierto que no hay un exceso de consumo ni producción, al menos no de la manera típica de concebirse, pero sí que es posible optimizar el tiempo y el esfuerzo invertido en las clases por parte de los profesores si se optimizan los horarios o se reduce el tiempo diario que permanece abierta la academia gracias a esta optimización.