IES RIBERA DE CASTILLA

PROYECTO SGE

CFGS Desarrollo de Aplicaciones Multiplataforma
Informática y Comunicaciones

Desarrollo del módulo “manage” con Odoo ERP; para gestionar proyectos usando metodologías ágiles: scrum

Año: *2024/2025*

Fecha de presentación: *22/12/2024*

Nombre y Apellidos: Rebeca Martínez de León
Email: rebeca.marleo@educa.jcyl.es

Índice

Modificar la **portada** con vuestros datos personales.

TÍTULO DEL PROYECTO: “Desarrollo del módulo “manage” con Odoo ERP; para gestionar proyectos usando metodologías ágiles: scrum”.

El primer apartado que vamos a incluir es la **INTRODUCCIÓN**: hablar brevemente sobre los sistemas ERP y las metodologías ágiles (en concreto sobre SCRUM).

Segundo apartado: **ORGANIZACIÓN DE LA MEMORIA** (lo vamos a dejar para el final; consistirá en enumerar los apartados de la memoria, con una pequeña descripción de cada uno de ellos).

Tercer apartado: **ESTADO DEL ARTE**. Con los siguientes subapartados:

1. ERP
    a. Definición de los ERP
    b. Evolución de los ERPs
    c. Principales ERP
    d. ERP seleccionado (Odoo)
    e. Instalación y desarrollo (*formas de instalación, explicando la que se va a usar para desarrollar el proyecto: Docker)*
    f. Especificaciones técnicas
        i. Arquitectura de Odoo
        ii. Composición de un módulo
2. SCRUM
    a. Definición de SCRUM
    b. Evolución
    c. Funcionamiento
    d. Principales conceptos *(explicar los principales conceptos: proyecto, historias de usuario, sprint, tarea…)*

Enlace sugerido: https://www.atlassian.com/es/agile/scrum

CONTINUACIÓN PROYECTO MANAGE

Investigar una posible ampliación de este proyecto e implementarla en vuestra aplicación. La ampliación tiene que incluir alguno(s) de los siguientes apartados:

- Algún aspecto no visto hasta ahora en el módulo de Sistemas de Gestión Empresarial
- Algún aspecto relacionado con el uso de CRUD, usando el ORM de Odoo

Añadir los siguientes puntos, a continuación de los que ya teníamos anteriormente en la memoria:

Descripción general del proyecto:

- **Objetivos** (breve descripción de lo que se ha pretendido alcanzar con el proyecto);
- **Entorno de trabajo** (explicar todas las herramientas utilizadas para desarrollar el proyecto: Docker, navegador, visual studio code…)

Diseño de la aplicación:

- **Modelo relacional de la BBDD**
- **Partes del proyecto** (models, views, security…), explicando brevemente lo que consideréis importante
- **Ampliación del proyecto,** explicando detalladamente el objetivo de la ampliación, el desarrollo…

Pruebas de funcionamiento

Conclusiones y posibles ampliaciones

Bibliografía

Normas de entrega:

Extensión mínima: 30 páginas

Formato entrega:

- **Repositorio de Github, con los archivos generados en el desarrollo.**
- **Archivo README con este contenido: título del proyecto, descripción del proyecto, enlace a la memoria del proyecto en formato pdf**
- **La memoria en archivo independiente,**
    - **formato pdf;**
    - **nombre: Apellido1_Apellido2_Nombre_proyectomanage.pdf**

Presentación oral en clase de algunos apartados del proyecto:

- **Descripción general del proyecto**
- **Diseño de la aplicación (incluir vuestra ampliación)**
- **Conclusiones y posibles ampliaciones**

Criterios calificación memoria proyecto manage:

Presentación formal (20%)

- Se ajusta a los requerimientos formales establecidos: portada, bibliografía, formato entrega, nombre archivo
- Se incluye un índice estructurado y coherente con el contenido del proyecto
- El texto está bien redactado, no presenta incoherencias gramaticales ni faltas de ortografía
- Presenta el proyecto en forma y plazo establecidos (1 punto menos por cada día de retraso)

Contenidos (40%)

- Originalidad del tema elegido (contenidos originales, no repetidos)

- Grado de dificultad en orden a la investigación de contenidos
- Grado de profundización en la investigación de contenidos
- Explicación clara y concisa, con capturas, explicaciones breves…
- Incluye todos los apartados requeridos en el proyecto

Defensa oral (40%)

- Se ajusta al tiempo marcado
- Objetivo del proyecto
- Puntos esenciales de la resolución
- Conclusiones finales
- Grado de conocimiento y dominio de los contenidos expuestos
- Lenguaje técnico utilizado

Introducción

Los sistemas ERP simplifican enormemente las tareas repetitivas que se deben realizar en una empresa. Su funcionamiento sencillo e intuitivo y su complejidad interna las convierte en las herramientas perfectas para el manejo de las labores del trabajo.

Por su lado, las metodologías ágiles consisten en la división de las tareas en distintos equipos enfocados en cada una de ellas de forma individual. El método Scrum sigue esta filosofía y está enfocado a la realización de tareas en cortos periodos de tiempo y con gran probabilidad de cambios a lo largo del proyecto. Es por esto que el método Scrum es conocido como “la metodología del caos”.

Organización de la memoria

La memoria está dividida en 3 apartados principales.

El primer apartado es la introducción. En él se presentan brevemente los sistemas ERP y las metodologías ágiles, en particular Scrum.

En el segundo apartado se resumen el resto de apartados.

Por último, en el tercer apartado se explica el estado del arte en cuestión de ERP y Scrum. Este apartado se divide a su vez en otros subapartados.

Primero se habla de los ERP, comenzando por una definición del término y continuando con su evolución. Después se habla de los principales ERP, del ERP seleccionado para realizar el proyecto y de la instalación y desarrollo en dicho ERP. Por último, se mencionan sus especificaciones técnicas, que son la arquitectura de Odoo y la composición de un módulo.

Luego se explica Scrum, también iniciando con una definición del mismo y su evolución a lo largo de los años. Además, se explica su funcionamiento y sus principales conceptos.

Estado del arte

1. ERP

a) Definición de los ERP

Un ERP o sistema de planificación de recursos empresariales es un conjunto de programas que integra todos los datos y procesos de una organización en un solo sistema unificado para la realización de las operaciones internas más importantes de la empresa de forma automática, permitiendo la trazabilidad de todos sus procesos y dando así paso a la planificación y optimización de los recursos.

b) Evolución de los ERPs

1960: En esta década se produjo el surgimiento de los MRP o sistemas de planificación de requerimientos de materiales, los cuales se utilizaban para la gestión de inventario y procesos de producción. Estos sistemas tenían grandes limitaciones, ya que estaban principalmente enfocados al sector manufacturero y su instalación requería una gran inversión en hardware. A pesar de ello, tuvieron un gran impacto debido a los beneficios que proporcionaban en términos de eficiencia y asentaron las bases de lo que estaría por venir.

1970: Durante los 70, los sistemas MRP que habían surgido en la década anterior se fueron perfeccionando. Así adquirieron nuevas funcionalidades como la planificación de la producción, la programación de la planta o la gestión de capacidades. Los MRP eran cada vez más populares y cada vez más empresas contaban con ellos.

1980: La década de los 80 fue el momento clave en que los MRP evolucionaron para dar lugar a los ERP. Estos nuevos sistemas contaban con características más allá de la planificación de la producción, como manejo de finanzas, recursos humanos, ventas o distribución. Los ERP supusieron una revolución para las empresas, ya que les permitieron tener una visión unificada de todos los procesos de su negocio.

1990: En los años 90 los ERP se expandieron a una gran variedad de industrias, incluyendo a medianas y pequeñas empresas, ya que durante esta década estos sistemas se volvieron más asequibles debido a la reducción del costo de hardware. Comenzó la creación de módulos cada vez más especializados para la labor de cada empresa. Durante este periodo se produjo también el surgimiento de internet y con él el inicio de la globalización, permitiendo que los ERP ayudasen a la coordinación de tareas a nivel global.

2000: Con la llegada del nuevo milenio se introdujeron los nuevos ERP en la nube, los cuales permitieron a las empresas acceder a sus sistemas de planificación desde internet, reduciendo enormemente los costos de infraestructuras y mantenimiento de las mismas y aumentando aún más su accesibilidad a las empresas de menor tamaño. Además, se implementó una nueva forma de adquirir estos servicios; en lugar de pagar el precio completo una sola vez se creó un modelo de suscripciones. En esta década se empezaron a integrar nuevas tecnologías como el big data, el internet de las cosas o la inteligencia artificial.

2010: Durante la década del 2010 las empresas solicitaron poder acceder a sus ERP desde cualquier parte por lo que se desarrollaron aplicaciones móviles. También hubo un enorme auge en la personalización de estos sistemas. Las empresas ya no se conformaban con tener soluciones genéricas y querían módulos adaptados a sus respectivas necesidades.

2020: En la actualidad, los ERP siguen creciendo con un fuerte enfoque en el uso de inteligencia artificial, el análisis de datos avanzado y la automatización de procesos. Estas tecnologías permiten a las empresas analizar información de forma masiva y predecir tendencias que les ayuden a la toma de decisiones. Es destacable además la mejora en las interfaces, siendo cada vez más intuitivas y personalizables para la mayor comodidad del usuario, incrementando así la productividad.

c) Principales ERP

En la actualidad contamos con un amplio abanico de sistemas ERP entre los cuales elegir, pero los principales son los siguientes:

- ERP Oracle Fusion Cloud
- ERP Microsoft Dynamics 365
- Workday Enterprise Management Cloud
- Oracle NetSuite
- SAP S/ 4HANA

d) ERP seleccionado

Entre los ERP disponibles, Odoo ha sido el ERP seleccionado para realizar este proyecto, ya que este se puede considerar una alternativa de código abierto muy sólida a sistemas ERP de alta categoría como Microsoft Dynamics o SAP.

e) Instalación y desarrollo

Para instalar Odoo debemos crearnos una cuenta en su página, después seleccionamos la versión de Odoo que deseemos y seguimos los pasos que nos indique el installer.

Para desarrollar un módulo en Odoo se utiliza el lenguaje Python. Gracias al comando scaffold podemos crear una estructura básica para nuestro proyecto que iremos modificando hasta crear el módulo personalizado de Odoo que queramos. Este comando se ejecuta en la terminal de Docker, el programa que emplearemos para manejar las bases de datos de nuestros módulos.

No existe un programa específico para programar módulos de Odoo por lo que podemos usar nuestro editor de texto de preferencia, en este caso Visual Studio Code.

f) Especificaciones técnicas

i. Arquitectura de Odoo

La arquitectura de Odoo está separada en varios niveles, siendo estos la presentación, la lógica empresarial y el almacenamiento de datos.

El nivel de la presentación es una combinación de HTML5, JavaScript y CSS.

El nivel lógico está escrito en Python puro.

El nivel de datos solo admite PostgreSQL.

ii. Composición de un módulo

Un módulo de Odoo está compuesto por los siguientes elementos, aunque ninguno de ellos es obligatorio:

- Objetos de negocio: son clases Python en las que se declara la información de los modelos del módulo.
- Vistas de objetos: define la interfaz de usuario.
- Archivos de información: los archivos XML o CSV que declaran los datos del modelo.
- Controladores web: manejan las solicitudes de los navegadores web.
- Datos web estáticos: las imágenes, archivos CSS o JavaScript utilizados por la interfaz web o el sitio web.

2. Scrum

a) Definición de Scrum

Scrum es un marco de gestión de proyectos ágil que ayuda a equipos a estructurar y gestionar el trabajo conjunto gracias a los valores, principios y prácticas característicos de la metodología, que anima a estos equipos a aprender de sus experiencias, a organizarse mientras trabajan en solventar un problema y a reflexionar tanto sobre sus victorias como sus derrotas y aprender de ambas.

b) Evolución

1986: es el año en el que nace esta metodología. Sus creadores, Ikujiro Nonaka e Hirotaka Takeuchi, se basaron en un estudio realizado en distintas empresas que estaban buscando un nuevo enfoque de trabajo. Estas empresas estaban basando su producción en equipos de trabajo autoorganizados y enfatizaba la importancia de esta autonomía.

1993: Jeff Sutherland y su equipo crean el proceso de Scrum para desarrollo de software combinando los conceptos del artículo de 1986 con los de desarrollo orientado a objetos, control de procesos empírico, desarrollo iterativo incremental, proceso de software y mejora de la productividad, así como el desarrollo de sistemas complejos y dinámicos.

1995: Jeff Sutherland y Ken Schwaber crean un conjunto de reglas y buenas prácticas enfocadas al desarrollo de software y lo bautizan con el nombre de Scrum.

2001: tras la publicación del Manifiesto Ágil la popularidad de Scrum aumentó.

2020: la guía Scrum recibe una actualización importante para hacer la práctica más accesible, sencilla y abierta para entornos fuera del campo de las tecnologías de la información.

c) Funcionamiento

La metodología Scrum consiste en abordar un proyecto dividiéndolo en sprints o partes más pequeñas. Dentro de este entorno de trabajo hay que seguir una serie de fases para abordar cada

tarea, y participan unos roles específicos que garantizan el cumplimiento de esta filosofía de trabajo.

d) Principales conceptos

Los principales conceptos sobre los que se fundamenta la metodología Scrum son los siguientes:

- Proyecto: es el trabajo general que el equipo va a desarrollar. Suelen planificarse completamente desde el principio y su estructura se define a través de las funcionalidades y las metas a cumplir.
- Sprints: son ciclos cortos y repetitivos que suelen durar entre 1 o 4 semanas. En cada sprint, el equipo se enfoca en completar un conjunto de tareas definidas para entregar una versión funcional del producto.
- Historias de usuario: son la manera en que Scrum representa los requisitos o funcionalidades del proyecto desde la perspectiva del usuario final. Normalmente se escriben en un formato específico y deben ser breves, así ayudan a enfocarse en las necesidades y beneficios reales del usuario.
- Tarea: es una división más pequeña de trabajo que deriva de una historia de usuario y representa un paso concreto necesario para completar esta. Deben ser muy específicas y breves, pues su duración óptima es de a penas unas horas o un día como máximo.
- Equipos auto-organizados y multifuncionales: son pequeños, de entre 5 a 9 personas y cada miembro aporta habilidades específicas, trabajando de manera colaborativa sin una jerarquía fija.

Descripción general del proyecto

Objetivos

El objetivo de este proyecto es crear un módulo de odoo que sirva para la organización del trabajo mediante la metodología ágil Scrum, permitiendo la creación y manejo de proyectos, sprints, historias, tecnologías, desarrolladores y tareas.

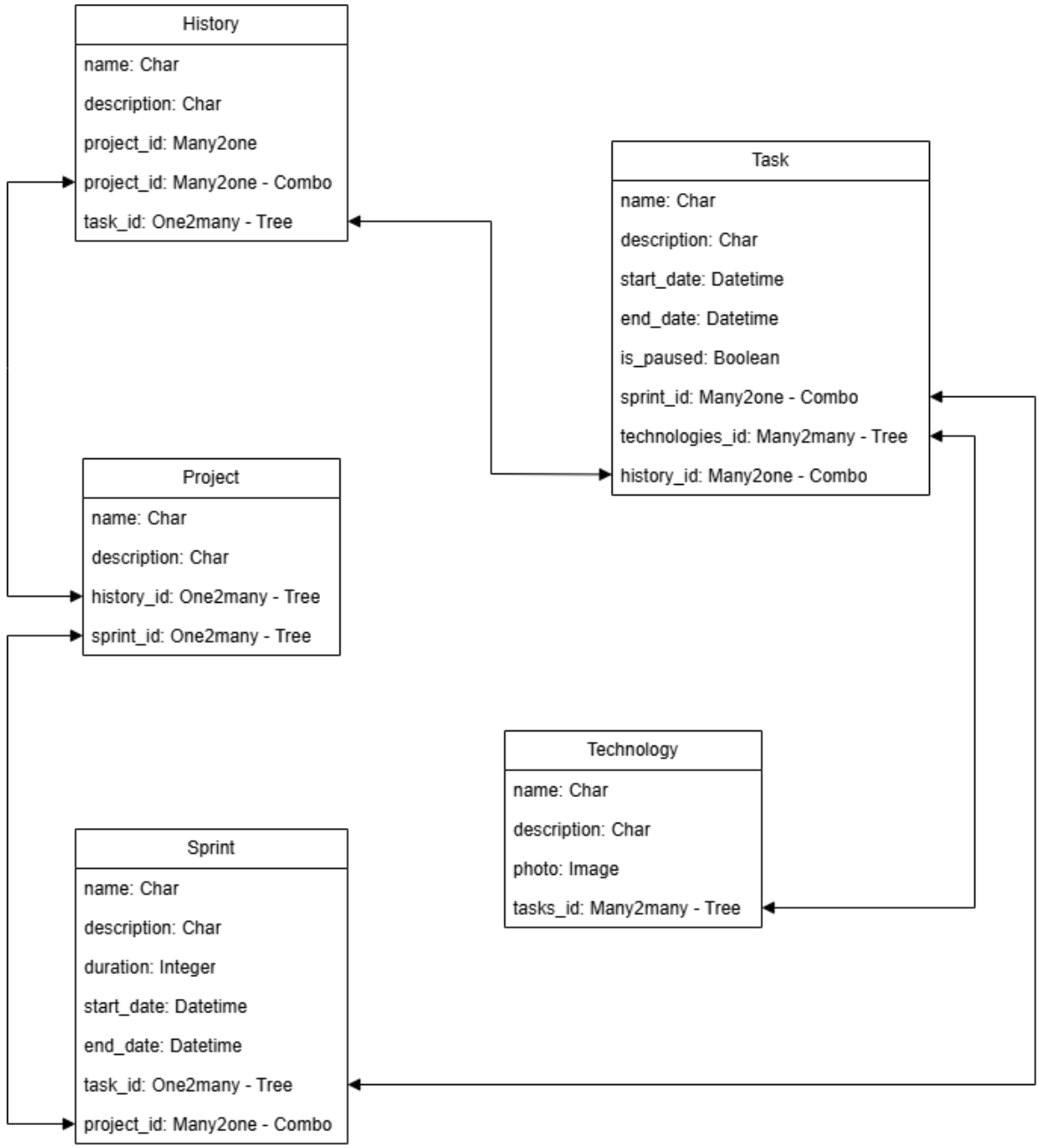
Entorno de trabajo

Para la realización del proyecto se utilizó:

- Visual Studio Code: para la creación y modificación de archivos .py, .xml y .csv.
- Google Chrome: como navegador en el que desplegar el localhost y comprobar los cambios realizados en el módulo.
- Docker: la plataforma para automatizar el despliegue del módulo dentro de los contenedores.

Diseño de la aplicación

Modelo relacional de la BBDD



Partes del proyecto

Modelos

- Modelo developer.py:

developer.py 1

models > developer.py > developer

```python
# -*- coding: utf-8 -*-

from odoo import models, fields


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

    is_dev = fields.Boolean(default = True)

    technologies = fields.Many2many("managerebeca.technology",
                                    relation="developer_technologies",
                                    column1="developer_id",
                                    column2="technologies_id")
```

El modelo developer.py hereda del módulo res.partner mediante su atributo _inherit.

```python
    _inherit = 'res.partner'
```

Tiene un atributo que se establece como True por defecto para ver si los nuevos developers añadidos desde su formulario correspondiente son desarrolladores.

```python
    is_dev = fields.Boolean(default = True)
```

Además, tiene una relación en la que un desarrollador puede usar una o más tecnologías diferentes y a su vez esas tecnologías pueden ser utilizadas por más de un desarrollador.

```python
    technologies = fields.Many2many("managerebeca.technology",
                                    relation="developer_technologies",
                                    column1="developer_id",
                                    column2="technologies_id")
```

- Modelo history.py:

history.py 1

models > history.py > ...

```python
# -*- coding: utf-8 -*-

from odoo import models, fields, api


class history(models.Model):
    _name = 'managerebeca.history'
    _description = 'managerebeca.history'

    name = fields.Char(String="Nombre")
    description = fields.Char(String="Descripción")

    project_id = fields.Many2one("managerebeca.project", string = "Proyecto", ondelete="set null")
    task_id = fields.One2many(string = "Tarea", comodel_name = "managerebeca.task", inverse_name = "history_id")
    used_technologies = fields.Many2many("managerebeca.technology", compute = "_get_used_technologies")

    def _get_used_technologies(self):
        for history in self:
            technologies = None # Array para concatenar todas las tecnologías. Inicialmente no tiene valor
            for task in history.tasks: # Para cada una de las tareas de la historia
                if not technologies:
                    technologies = task.technologies
                else:
                    technologies = technologies + task.technologies
            history.used_technologies = technologies # Asignar las tecnologías a la historia
```

El modelo history.py es un modelo nativo del módulo managerebeca.

Tiene una relación en la que una historia solo puede pertenecer a un proyecto pero un proyecto puede tener múltiples historias.

```python
    project_id = fields.Many2one("managerebeca.project", string = "Proyecto", ondelete="set null")
```

Además, una historia puede tener múltiples tareas pero una tarea solo puede pertenecer a una historia.

```python
    task_id = fields.One2many(string = "Tarea", comodel_name = "managerebeca.task", inverse_name = "history_id")
```

En una historia se pueden utilizar múltiples tecnologías diferentes y a su vez cada una de esas tecnologías pueden ser utilizadas en muchas tareas diferentes. Este campo se autocompleta mediante el método _get_used_technologies, mediante el cual se recogen todas las tecnologías usadas en las tareas en una lista.

```python
    used_technologies = fields.Many2many("managerebeca.technology", compute = "_get_used_technologies")
```

```python
    def _get_used_technologies(self):
        for history in self:
            technologies = None # Array para concatenar todas las tecnologías. Inicialmente no tiene valor
            for task in history.tasks: # Para cada una de las tareas de la historia
                if not technologies:
                    technologies = task.technologies
                else:
                    technologies = technologies + task.technologies
            history.used_technologies = technologies # Asignar las tecnologías a la historia
```

- Modelo project.py:

project.py 1

models > project.py > project

```python
# -*- coding: utf-8 -*-

from odoo import models, fields, api


class project(models.Model):
    _name = 'managerebeca.project'
    _description = 'managerebeca.project'

    name = fields.Char(String="Nombre")
    description = fields.Char(String="Descripción")

    history_ids = fields.One2many(string = "Historia", comodel_name = "managerebeca.history", inverse_name = "project_id")
    sprints_id = fields.One2many(string = "Sprint", comodel_name = "managerebeca.sprint", inverse_name = "project_id")
```

El modelo project.py es un modelo nativo del módulo managerebeca.

Tiene una relación en la que un proyecto puede tener múltiples historias pero una historia solo puede pertenecer a un proyecto.

```python
    history_ids = fields.One2many(string = "Historia", comodel_name = "managerebeca.history", inverse_name = "project_id")
```

Además, un proyecto puede tener múltiples sprints pero un sprint solo puede pertenecer a un proyecto.

```python
    sprints_id = fields.One2many(string = "Sprint", comodel_name = "managerebeca.sprint", inverse_name = "project_id")
```

- Modelo sprint.py:

sprint.py 1

models > sprint.py > sprint > _get_end_date

```python
# -*- coding: utf-8 -*-

from odoo import models, fields, api
import datetime


class sprint(models.Model):
    _name = 'managerebeca.sprint'
    _description = 'managerebeca.sprint'

    name = fields.Char(string="Nombre")
    description = fields.Text(string="Descripción")
    duration = fields.Integer(default = 15)
    start_date = fields.Datetime(string="Fecha inicio")
    end_date = fields.Datetime(string="Fecha fin", compute="_get_end_date", store=True)

    tasks_ids = fields.One2many(string = "Tareas", comodel_name = "managerebeca.task", inverse_name = "sprint_id")
    project_id = fields.Many2one("managerebeca.project")

    @api.depends('start_date', 'duration')
    def _get_end_date(self):
        for sprint in self:
            #try:
            if isinstance(sprint.start_date, datetime.datetime) and sprint.duration > 0:
                sprint.end_date = sprint.start_date + datetime.timedelta(days=sprint.duration)
            else:
                sprint.end_date = sprint.start_date
```

El modelo sprint.py es un modelo nativo del módulo managerebeca.

Tiene un atributo que se establece como 15 por defecto para establecer la duración base de un sprint que luego podrá ser modificada.

```
13      duration = fields.Integer(default = 15)
```

Su atributo start_date determina la fecha de inicio del sprint y su atributo end_date determina la fecha de finalización. Este útlimo se calcula a partir de la función _get_end_date, que toma la fecha de inicio y la duración y las suma para obtener la fecha de finalización.

```
14      start_date = fields.Datetime(string="Fecha inicio")
```

```
15      end_date = fields.Datetime(string="Fecha fin", compute="_get_end_date", store=True)
```

```
20      @api.depends('start_date', 'duration')
21      def _get_end_date(self):
22          for sprint in self:
23              #try:
24              if isinstance(sprint.start_date, datetime.datetime) and sprint.duration > 0:
25                  sprint.end_date = sprint.start_date + datetime.timedelta(days=sprint.duration)
26              else:
27                  sprint.end_date = sprint.start_date
```

Tiene una relación en la que un sprint puede tener múltiples tareas pero una tarea solo puede pertenecer a un sprint.

```
17      tasks_ids = fields.One2many(string = "Tareas", comodel_name = "managerebeca.task", inverse_name = "sprint_id")
```

Además, tiene una relación en la que un sprint solo puede pertenecer a un proyecto pero un proyecto puede tener múltiples sprints.

```
18      project_id = fields.Many2one("managerebeca.project")
```

- Modelo task.py:

task.py 1 ×

models > task.py > ...

```python
# -*- coding: utf-8 -*-

import datetime
from odoo import models, fields, api


class task(models.Model):
    _name = 'managerebeca.task'
    _description = 'managerebeca.task'

    name = fields.Char(String="Nombre")
    description = fields.Char(String="Descripción")
    start_date = fields.Datetime()
    end_date = fields.Datetime()
    is_paused = fields.Boolean()

    sprint_id = fields.Many2one("managerebeca.sprint", compute = "_get_sprint", store = True)
    technologies_ids = fields.Many2many(comodel_name = "managerebeca.technology",
                                        relation = "technologies_tasks",
                                        column1 = "technologies_ids",
                                        column2 = "tasks_ids",
                                        string = "Tecnologías",)
    history_id = fields.Many2one("managerebeca.history", ondelete="set null", help="Historia relacionada")
    definition_date = fields.Datetime(default=lambda p: datetime.datetime.now())
    project_id = fields.Many2one("managerebeca.project", related = "history_id.project_id", readonly = True)
    code = fields.Char(String = "Código", compute = "_get_code")

    @api.depends('start_date', 'end_date')
    def _get_sprint(self):
        for task in self:
            sprints = self.env['managerebeca.sprint'].search([('project_id', '=', task.history_id.project_id.id)])
            found = False
            for sprint in sprints:
                if isinstance(sprint.end_date, datetime.datetime) and sprint.end_date > datetime.datetime.now():
                    task.sprint_id = sprint.id
                    found = True
            if not found:
                task.sprint_id = False

    #@api.one
    def _get_code(self):
        for task in self:
            # try:
                task.code = "TSK_"+str(task.id)
                #_logger.info("Código generado: "+task.code)
            #except:
                #raise ValidationError(_("Generación de código errónea"))
```

El modelo task.py es un modelo nativo del módulo managerebeca.

Tiene una relación en la que una tarea solo puede pertenecer a un sprint pero un sprint puede tener múltiples tareas. En este campo se utiliza la función _get_sprint para obtener los sprints que tengan una fecha de finalización posterior a la fecha actual. De esta manera se valida que las tareas no se puedan asociar a sprints que ya hayan terminado.

```python
    sprint_id = fields.Many2one("managerebeca.sprint", compute = "_get_sprint", store = True)
```

```python
@api.depends('start_date', 'end_date')
def _get_sprint(self):
    for task in self:
        sprints = self.env['managerebeca.sprint'].search([('project_id', '=', task.history_id.project_id.id)])
        found = False
        for sprint in sprints:
            if isinstance(sprint.end_date, datetime.datetime) and sprint.end_date > datetime.datetime.now():
                task.sprint_id = sprint.id
                found = True
        if not found:
            task.sprint_id = False
```

Además, una tarea puede usar una o más tecnologías diferentes y a su vez esas tecnologías pueden ser utilizadas por más de una tarea.

```python
technologies_ids = fields.Many2many(comodel_name = "managerebeca.technology",
                                    relation = "technologies_tasks",
                                    column1 = "technologies_ids",
                                    column2 = "tasks_ids",
                                    string = "Tecnologías",)
```

También tiene una relación en la que una tarea solo puede pertenecer a una historia pero una historia puede tener múltiples tareas.

```python
history_id = fields.Many2one("managerebeca.history", ondelete="set null", help="Historia relacionada")
```

Tiene un atributo que se establece por defecto como la fecha y hora actuales que define la hora de creación de la tarea.

```python
definition_date = fields.Datetime(default=lambda p: datetime.datetime.now())
```

Además, tiene una relación en la que una tarea solo puede pertenecer a un proyecto pero un proyecto puede tener múltiples tareas.

```python
project_id = fields.Many2one("managerebeca.project", related = "history_id.project_id", readonly = True)
```

También tiene un atributo que establece un código a la tarea. Este atributo se calcula automáticamente mediante el método _get_code que establece el código como una concatenación de la cadena TSK_ y el id correspondiente de dicha tarea.

```python
code = fields.Char(String = "Código", compute = "_get_code")
```

```python
#@api.one
def _get_code(self):
    for task in self:
        # try:
            task.code = "TSK_"+str(task.id)
            #_logger.info("Código generado: "+task.code)
        #except:
            #raise ValidationError(_("Generación de código errónea"))
```

- Modelo technology.py:

technology.py 1

models > technology.py > ...

```python
# -*- coding: utf-8 -*-

from odoo import models, fields, api


class technology(models.Model):
    _name = 'managerebeca.technology'
    _description = 'managerebeca.technology'

    name = fields.Char(String="Nombre")
    description = fields.Char(String="Descripción")
    photo = fields.Image(string="Imagen", max_width=200, max_height=200)

    tasks_ids = fields.Many2many(comodel_name = "managerebeca.task",
                                 relation = "technologies_tasks",
                                 column1 = "tasks_ids",
                                 column2 = "technologies_ids")
```

El modelo technology.py es un modelo nativo del módulo managerebeca.

Tiene un atributo photo para introducir una foto de la tecnología empleada.

```python
    photo = fields.Image(string="Imagen", max_width=200, max_height=200)
```

Tiene una relación en la que una tecnología puede usarse en una o más tareas diferentes y a su vez esas tareas pueden ser utilizadas por más de una tecnología.

```python
    tasks_ids = fields.Many2many(comodel_name = "managerebeca.task",
                                 relation = "technologies_tasks",
                                 column1 = "tasks_ids",
                                 column2 = "technologies_ids")
```

Vistas

- Vista developer.xml:

developer.xml

views > developer.xml > odoo

```xml
<odoo>
    <data>
        <record model="ir.ui.view" id="managerebeca.devs_partner_form">
            <field name="name">manage devs form</field>
            <field name="model">res.partner</field>
            <field name="inherit_id" ref="base.view_partner_form"></field>
            <field name="mode">primary</field>
            <field name="arch" type="xml">
                <xpath expr="//sheet/notebook/page[@name='internal_notes']" position="after">
                    <page name="devs" string="Devs">
                        <group>
                            <group>
                                <field name="technologies"></field>
                            </group>
                        </group>
                    </page>
                </xpath>
            </field>
        </record>

        <record model="ir.actions.act_window" id="managerebeca.action_developer_window">
            <field name="name">manage developer window</field>
            <field name="res_model">res.partner</field>
            <field name="view_mode">tree,form</field>
        </record>

        <record model="ir.actions.act_window.view" id="managerebeca.action_view_developer_tree">
            <field name="sequence" eval="1"></field>
            <field name="view_mode">tree</field>
            <field name="view_id" ref="base.view_partner_tree"></field>
            <field name="act_window_id" ref="managerebeca.action_developer_window"></field>
        </record>

        <record model="ir.actions.act_window.view" id="managerebeca.action_view_developer_form">
            <field name="sequence" eval="2"></field>
            <field name="view_mode">form</field>
            <field name="view_id" ref="managerebeca.devs_partner_form"></field>
            <field name="act_window_id" ref="managerebeca.action_developer_window"></field>
        </record>

        <menuitem name="Devs" id="menu_managerebeca_developer" parent="menu_managerebeca_management"
            action="managerebeca.action_developer_window" />
    </data>
</odoo>
```

La vista developer.xml hereda de la vista del formulario y la vista tree del módulo res.partner mediante el atributo ref=”base.view_partner_form” y ref=”base.view_partner_tree” respectivamente.

```xml
6        <field name="inherit_id" ref="base.view_partner_form"></field>
```

```xml
30       <field name="view_id" ref="base.view_partner_tree"></field>
```

La vista developer.xml se puede acceder mediante su id menu_managerebeca_developer que, al igual que todas las vistas del módulo, cuelga del padre menu_managerebeca_management.

```
41          <menuitem name="Devs" id="menu_managerebeca_developer" parent="menu_managerebeca_management"
```

- Vista history.xml:

history.xml

views > history.xml > odoo

```xml
<odoo>
    <data>
        <!-- explicit list view definition -->
        <record model="ir.ui.view" id="vista_managerebeca_history_tree">
            <field name="name">vista_managerebeca_history_tree</field>
            <field name="model">managerebeca.history</field>
            <field name="arch" type="xml">
                <tree>
                    <field name="name" />
                    <field name="description" />
                </tree>
            </field>
        </record>

        <record model="ir.ui.view" id="vista_managerebeca_history_form">
            <field name="name">vista_managerebeca_history_form</field>
            <field name="model">managerebeca.history</field>
            <field name="arch" type="xml">
                <form string="formulario_history">
                    <sheet>
                        <group name="group_top">
                            <field name="name" />
                            <field name="description" />
                            <field name="project_id" />
                            <field name="task_id" />
                            <field name="used_technologies" />
                        </group>
                    </sheet>
                </form>
            </field>
        </record>

        <record model="ir.actions.act_window" id="accion_managerebeca_history_form">
            <field name="name">Registros del historial</field>
            <field name="type">ir.actions.act_window</field>
            <field name="res_model">managerebeca.history</field>
            <field name="view_mode">tree,form</field>
            <field name="help" type="html">
                <p class="oe_view_nocontent_create">
                    Historial
                </p>
                <p>Click <strong> 'Crear' </strong> para añadir nuevos elementos
                </p>
            </field>
        </record>

        <!-- actions -->
```

```
49          <menuitem name="Histories" id="menu_managerebeca_history" parent="menu_managerebeca_management"
50              action="accion_managerebeca_history_form" />
51
52      </data>
53  </odoo>
```

La vista tree toma el modelo history de managerebeca y muestra los campos del nombre y la descripción de la historia.

```
6                   <field name="model">managerebeca.history</field>
```

```
9                       <field name="name" />
10                      <field name="description" />
```

La vista form toma el modelo history de managerebeca y recibe información para los campos nombre, descripción, proyecto, tareas y tecnologías usadas.

```
17                  <field name="model">managerebeca.history</field>
```

```
22                      <field name="name" />
23                      <field name="description" />
24                      <field name="project_id" />
25                      <field name="task_id" />
26                      <field name="used_technologies" />
```

La vista history.xml se puede acceder mediante su id menu_managerebeca_history que, al igual que todas las vistas del módulo, cuelga del padre menu_managerebeca_management.

```
49          <menuitem name="Histories" id="menu_managerebeca_history" parent="menu_managerebeca_management"
```

- Vista project.xml:

project.xml

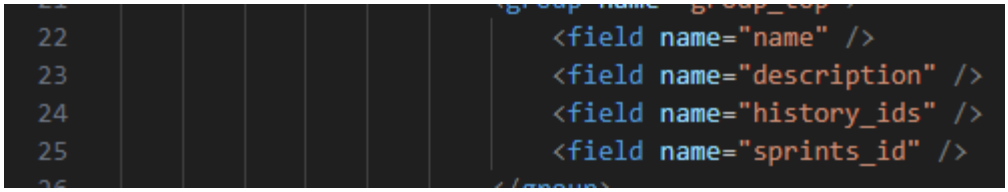
views > project.xml > odoo

```xml
<odoo>
    <data>
        <!-- explicit list view definition -->
        <record model="ir.ui.view" id="vista_managerebeca_project_tree">
            <field name="name">vista_managerebeca_project_tree</field>
            <field name="model">managerebeca.project</field>
            <field name="arch" type="xml">
                <tree>
                    <field name="name" />
                    <field name="description" />
                </tree>
            </field>
        </record>

        <record model="ir.ui.view" id="vista_managerebeca_project_form">
            <field name="name">vista_managerebeca_project_form</field>
            <field name="model">managerebeca.project</field>
            <field name="arch" type="xml">
                <form string="formulario_project">
                    <sheet>
                        <group name="group_top">
                            <field name="name" />
                            <field name="description" />
                            <field name="history_ids" />
                            <field name="sprints_id" />
                        </group>
                    </sheet>
                </form>
            </field>
        </record>

        <record model="ir.actions.act_window" id="accion_managerebeca_project_form">
            <field name="name">Listado de proyectos</field>
            <field name="type">ir.actions.act_window</field>
            <field name="res_model">managerebeca.project</field>
            <field name="view_mode">tree,form</field>
            <field name="help" type="html">
                <p class="oe_view_nocontent_create">
                    Listado de proyectos
                </p>
                <p>Click <strong> 'Crear' </strong> para añadir nuevos elementos
                </p>
            </field>
        </record>

        <!-- actions -->

        <menuitem name="Projects" id="menu_managerebeca_project" parent="menu_managerebeca_management"
            action="accion_managerebeca_project_form" />

    </data>
</odoo>
```

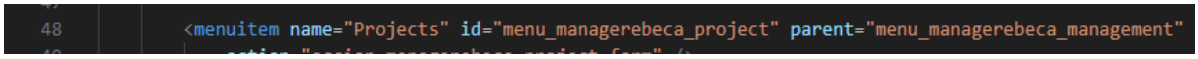
La vista tree toma el modelo project de managerebeca y muestra los campos del nombre y la descripción del proyecto.

```xml
9                    <field name="name" />
10                   <field name="description" />
```

La vista form toma el modelo project de managerebeca y recibe información para los campos nombre, descripción, historias y sprints.



La vista project.xml se puede acceder mediante su id menu_managerebeca_project que, al igual que todas las vistas del módulo, cuelga del padre menu_managerebeca_management.



- Vista sprint.xml:

sprint.xml

views > sprint.xml > odoo

```xml
<odoo>
    <data>
        <!-- explicit list view definition -->
        <record model="ir.ui.view" id="vista_managerebeca_sprint_tree">
            <field name="name">vista_managerebeca_sprint_tree</field>
            <field name="model">managerebeca.sprint</field>
            <field name="arch" type="xml">
                <tree>
                    <field name="name" />
                    <field name="description" />
                </tree>
            </field>
        </record>

        <record model="ir.ui.view" id="vista_managerebeca_sprint_form">
            <field name="name">vista_managerebeca_sprint_form</field>
            <field name="model">managerebeca.sprint</field>
            <field name="arch" type="xml">
                <form string="formulario_sprint">
                    <sheet>
                        <group name="group_top">
                            <field name="name" />
                            <field name="description" />
                            <field name="duration" />
                            <field name="start_date" />
                            <field name="end_date" />
                            <field name="tasks_ids" />
                            <field name="project_id" />
                        </group>
                    </sheet>
                </form>
            </field>
        </record>

        <record model="ir.actions.act_window" id="accion_managerebeca_sprint_form">
            <field name="name">Listado de sprints</field>
            <field name="type">ir.actions.act_window</field>
            <field name="res_model">managerebeca.sprint</field>
            <field name="view_mode">tree,form</field>
            <field name="help" type="html">
                <p class="oe_view_nocontent_create">
                    Listado de sprints
                </p>
                <p>Click <strong> 'Crear' </strong> para añadir nuevos elementos
                </p>
            </field>
        </record>

```

```
49          <!-- actions -->
50
51          <menuitem name="Sprints" id="menu_managerebeca_sprint" parent="menu_managerebeca_management"
52              action="accion_managerebeca_sprint_form" />
53
54      </data>
55  </odoo>
```

La vista tree toma el modelo sprint de managerebeca y muestra los campos del nombre y la descripción del sprint.

```
 9                  <field name="name" />
10                  <field name="description" />
```

La vista form toma el modelo sprint de managerebeca y recibe información para los campos nombre, descripción, duración, fecha de inicio, fecha de finalización, tareas y proyecto.

```
22                      <field name="name" />
23                      <field name="description" />
24                      <field name="duration" />
25                      <field name="start_date" />
26                      <field name="end_date" />
27                      <field name="tasks_ids" />
28                      <field name="project_id" />
```

La vista sprint.xml se puede acceder mediante su id menu_managerebeca_sprint que, al igual que todas las vistas del módulo, cuelga del padre menu_managerebeca_management.

```
51          <menuitem name="Sprints" id="menu_managerebeca_sprint" parent="menu_managerebeca_management"
```

- Vista task.xml:

task.xml

views > task.xml > odoo

```xml
<odoo>
    <data>
        <!-- explicit list view definition -->
        <record model="ir.ui.view" id="vista_managerebeca_task_tree">
            <field name="name">vista_managerebeca_task_tree</field>
            <field name="model">managerebeca.task</field>
            <field name="arch" type="xml">
                <tree>
                    <field name="name" />
                    <field name="description" />
                </tree>
            </field>
        </record>

        <record model="ir.ui.view" id="vista_managerebeca_task_form">
            <field name="name">vista_managerebeca_task_form</field>
            <field name="model">managerebeca.task</field>
            <field name="arch" type="xml">
                <form string="formulario_task">
                    <sheet>
                        <group name="group_top">
                            <field name="code" />
                            <field name="name" />
                            <field name="description" />
                            <field name="start_date" />
                            <field name="end_date" />
                            <field name="is_paused" />
                            <field name="sprint_id" />
                            <field name="history_id" />
                            <field name="technologies_ids" />
                            <field name="definition_date" />
                        </group>
                    </sheet>
                </form>
            </field>
        </record>

        <record model="ir.actions.act_window" id="accion_managerebeca_task_form">
            <field name="name">Listado de tareas</field>
            <field name="type">ir.actions.act_window</field>
            <field name="res_model">managerebeca.task</field>
            <field name="view_mode">tree,form</field>
            <field name="help" type="html">
                <p class="oe_view_nocontent_create">
                    Listado de tareas
                </p>
                <p>Click <strong> 'Crear' </strong> para añadir nuevos elementos
                </p>
```

```
49              </field>
50          </record>
51
52          <!-- actions -->
53
54          <menuitem name="Tasks" id="menu_managerebeca_task" parent="menu_managerebeca_management"
55              action="accion_managerebeca_task_form" />
56
57      </data>
58  </odoo>
```

La vista tree toma el modelo task de managerebeca y muestra los campos del nombre y la descripción de la tarea.

```
 9                      <field name="name" />
10                      <field name="description" />
```

La vista form toma el modelo task de managerebeca y recibe información para los campos código, nombre, descripción, fecha de inicio, fecha de finalización, en pausa, sprint, historia, tecnologías y fecha de creación.

```
22                      <field name="code" />
23                      <field name="name" />
24                      <field name="description" />
25                      <field name="start_date" />
26                      <field name="end_date" />
27                      <field name="is_paused" />
28                      <field name="sprint_id" />
29                      <field name="history_id" />
30                      <field name="technologies_ids" />
31                      <field name="definition_date" />
```

La vista task.xml se puede acceder mediante su id menu_managerebeca_task que, al igual que todas las vistas del módulo, cuelga del padre menu_managerebeca_management.

```
54          <menuitem name="Tasks" id="menu_managerebeca_task" parent="menu_managerebeca_management"
```

- Vista technology.xml:

technology.xml

views > technology.xml > odoo > data > record > field > p

```xml
<odoo>
    <data>
        <!-- explicit list view definition -->
        <record model="ir.ui.view" id="vista_managerebeca_technology_tree">
            <field name="name">vista_managerebeca_technology_tree</field>
            <field name="model">managerebeca.technology</field>
            <field name="arch" type="xml">
                <tree>
                    <field name="name" />
                    <field name="description" />
                </tree>
            </field>
        </record>

        <record model="ir.ui.view" id="vista_managerebeca_technology_form">
            <field name="name">vista_managerebeca_technology_form</field>
            <field name="model">managerebeca.technology</field>
            <field name="arch" type="xml">
                <form string="formulario_technology">
                    <sheet>
                        <group name="group_top">
                            <field name="name" />
                            <field name="description" />
                            <field name="photo" />
                            <field name="tasks_ids" />
                        </group>
                    </sheet>
                </form>
            </field>
        </record>

        <record model="ir.actions.act_window" id="accion_managerebeca_technology_form">
            <field name="name">Listado de tecnologías</field>
            <field name="type">ir.actions.act_window</field>
            <field name="res_model">managerebeca.technology</field>
            <field name="view_mode">tree,form</field>
            <field name="help" type="html">
                <p class="oe_view_nocontent_create">
                    Listado de tecnologías
                </p>
                <p>Click <strong> 'Crear' </strong> para añadir nuevos elementos
                </p>
            </field>
        </record>

        <!-- Top menu item -->

        <menuitem name="Manage de Rebeca" id="menu_managerebeca_raiz" />
```

```
49
50          <!-- menu categories -->
51
52          <menuitem name="Management" id="menu_managerebeca_management" parent="menu_managerebeca_raiz" />
53
54          <!-- actions -->
55
56          <menuitem name="Technologies" id="menu_managerebeca_technology" parent="menu_managerebeca_management"
57              action="accion_managerebeca_technology_form" />
58
59      </data>
60  </odoo>
```

La vista tree toma el modelo technology de managerebeca y muestra los campos del nombre y la descripción de la tecnología.

```
9                       <field name="name" />
10                      <field name="description" />
```

La vista form toma el modelo tecnología de managerebeca y recibe información para los campos nombre, descripción, foto y tareas.

```
22                      <field name="name" />
23                      <field name="description" />
24                      <field name="photo" />
25                      <field name="tasks_ids" />
```

La vista technology.xml se puede acceder mediante su id menu_managerebeca_technology que, al igual que todas las vistas del módulo, cuelga del padre menu_managerebeca_management.

```
56          <menuitem name="Technologies" id="menu_managerebeca_technology" parent="menu_managerebeca_management"
```

En esta vista se establece el padre menu_managerebeca_management del que cuelgan todas las vistas. Este es a su vez hijo de menu_management_raiz.

```
52          <menuitem name="Management" id="menu_managerebeca_management" parent="menu_managerebeca_raiz" />
```

El padre de menu_managerebeca_management.

```
48          <menuitem name="Manage de Rebeca" id="menu_managerebeca_raiz" />
```

Archivos adicionales

- Archivo __manifest__.py:

```python
# -*- coding: utf-8 -*-
{
    'name': "managerebeca",

    'summary': """
        Short (1 phrase/line) summary of the module's purpose, used as
        subtitle on modules listing or apps.openerp.com""",

    'description': """
        Long description of module's purpose
    """,

    'author': "My Company",
    'website': "https://www.yourcompany.com",

    # Categories can be used to filter modules in modules listing
    # Check https://github.com/odoo/odoo/blob/16.0/odoo/addons/base/data/ir_module_category_data.xml
    # for the full list
    'category': 'Uncategorized',
    'version': '0.1',

    # any module necessary for this one to work correctly
    'depends': ['base'],

    # always loaded
    'data': [
        'security/ir.model.access.csv',
        'views/technology.xml',
        'views/project.xml',
        'views/history.xml',
        'views/sprint.xml',
        'views/task.xml',
        'views/developer.xml',
        'views/views.xml',
        'views/templates.xml',

    ],
    # only loaded in demonstration mode
    'demo': [
        'demo/demo.xml',
    ],
}

```

En el archivo manifest se deben incluir los enlaces a cada una de las vistas y al archivo de seguridad. Es importante empezar por el archivo de seguridad ir.model.access.csv para que todas las vistas puedan cargar correctamente. Después se coloca el enlace a la vista technology ya que en ella se declaran los padres de los que descienden el resto de vistas.

```
        'security/ir.model.access.csv',
        'views/technology.xml',
        'views/project.xml',
        'views/history.xml',
        'views/sprint.xml',
        'views/task.xml',
        'views/developer.xml',
        'views/views.xml',
        'views/templates.xml',
```

- Archivo __init__.py:

```
_init_.py
models > _init_.py
# -*- coding: utf-8 -*-

from . import models
from . import task
from . import sprint
from . import project
from . import history
from . import technology
from . import developer
```

En el archivo init se deben incluir los enlaces a cada uno de los modelos.

- Archivo ir.model.access.csv:

```
ir.model.access.csv
security > ir.model.access.csv
id,name,model_id:id,group_id:id,perm_read,perm_write,perm_create,perm_unlink
access_managerebeca_technology,managerebeca.technology,model_managerebeca_technology,base.group_user,1,1,1,1
access_managerebeca_project,managerebeca.project,model_managerebeca_project,base.group_user,1,1,1,1
access_managerebeca_history,managerebeca.history,model_managerebeca_history,base.group_user,1,1,1,1
access_managerebeca_sprint,managerebeca.sprint,model_managerebeca_sprint,base.group_user,1,1,1,1
access_managerebeca_task,managerebeca.task,model_managerebeca_task,base.group_user,1,1,1,1
```

El archivo de seguridad que determina los permisos que tiene cada uno de los modelos presentes. Todos los modelos presentes deben estar incluidos en este archivo para funcionar correctamente.

Ampliación del proyecto

Aún no está hecho

Pruebas de funcionamiento

Conclusiones y posibles ampliaciones

Bibliografía

https://asana.com/es/resources/what-is-scrum

https://www.atlassian.com/es/agile/scrum

https://proyectosagiles.org/historia-de-scrum/

https://ilixum.com/evolucion-de-scrum/

https://entersol.com.mx/evolucion-sistemas-erp

https://wautechnologies.com/noticias/ranking-erp-mas-usados/

https://grupoaspasia.com/es/glosario/metodologia-scrum/

https://caroli.org/es/scrum-significado-aplicacion-conceptos-y-ejemplos/

https://code.visualstudio.com/

https://www.odoo.com/es_ES

https://www.docker.com/

https://ni.itc.services/blog/conoce-odoo-7/capitulo-1-descripcion-general-de-la-arquitectura-152

https://www.odoo.com/es_ES/page/download

https://ilimit.com/blog/metodologia-scrum/