

TECNOLÓGICO NACIONAL DE MÉXICO



INSTITUTO TECNOLÓGICO SUPERIOR DE APATZINGÁN



Reporte de Investigación

Taller de Programación Web II

Unidad II

Profesor:

I.S.C. Javier Cisneros Lucatero

Alumnos:

Pedro Cabello Mondragón

19020085

Félix Enrique Chavez Navarro

19020270

Eduardo Armenta López

19020292

Ramón Tarelo Cerna

19020365

Fecha: 25/06/2023

Índice:

1. Introducción.....	2
2. Desarrollo.....	2
2.1 Manifest.....	3
2.1.1 Explicación Manifest.....	4
2.2 Modelos.....	4
2.2.1 Explicación Modelos.....	5
2.3 Init.....	19
2.4 Views.....	20
2.4.1 Explicación Views:.....	22
2.5 Visualización:.....	23
3 Conclusión.....	40
4 Referencias:.....	41

1. Introducción

En este reporte aprenderemos cómo crear un módulo en odoo y cómo utilizarlo en el ambiente local, este proyecto fue elaborado en las horas de clase y trata sobre la gestión de equipos de fútbol, permite hacer las operaciones básicas de un CRUD en este framework. Este proyecto mostrará la forma en la que se configura el entorno de odoo así como la estructura de un proyecto desde los archivos necesarios para importar los modelos y cómo implementar las funcionalidades en el framework.

Aprenderemos la estructura que debe tener el backend que es toda la parte lógica del programa donde tendremos los archivos python que se encargaran de la funcionalidad y la parte del frontend donde tendremos nuestros archivos xml los que contendrán los archivos de las vistas.

utilizar edoo es una herramienta muy completa en el ámbito empresarial ya que este framework cuenta con una variedad de recursos que pueden favorecer a la optimización de trabajos en las empresas.

2. Desarrollo

Programar en Odoo implica utilizar el lenguaje de programación Python y el framework de desarrollo de Odoo para personalizar y extender las funcionalidades del sistema. A continuación, se muestra una guía básica para comenzar a programar en Odoo:

1. Configuración del entorno de desarrollo:
 - Instala Python en tu sistema.

- Instalar Odoo. Puedes descargarlo desde el sitio oficial de Odoo o utilizar la versión de código abierto disponible en GitHub.
 - Configura un entorno virtual para trabajar con Odoo. Esto te permitirá mantener tus proyectos de desarrollo separados y evitar conflictos con otras dependencias de Python.
2. Comprender la estructura de Odoo:
 - Odoo sigue una arquitectura basada en módulos. Cada funcionalidad se organiza en módulos individuales.
 - Los módulos de Odoo están escritos en Python y siguen una estructura de directorios específica.
 3. Creación de un nuevo módulo:
 - Crea un nuevo directorio para tu módulo en la ubicación adecuada dentro de la estructura de directorios de Odoo.
 - Dentro del directorio de tu módulo, crea los archivos necesarios, como `__init__.py`, `__manifest__.py`, y otros archivos de código fuente en Python.
 4. Desarrollo del módulo:
 - Define tus modelos: Los modelos de Odoo se definen como clases en Python y representan las entidades con las que trabajarás en tu módulo.
 - Define vistas: Las vistas determinan cómo se presentan los datos en Odoo. Puedes crear vistas XML para mostrar y editar registros de tus modelos.
 - Agrega lógica de negocio: Puedes implementar métodos en tus modelos para agregar lógica personalizada y realizar operaciones específicas.

2.1 Manifest

```
{
  'name': 'Teams',
  'version': '1.0',
  'category': 'Academica',
  'description': 'Control de equipos',
  'author': 'Yo',
  'maintainer': 'Yo',
  'website': 'http://www.itsa.edu.mx',
  'depends': ['base'],
  'data': [],
  'installable': True,
  'auto_install': False,
}
```

Este es el archivo manifest.py en donde se especifican las cosas relacionadas con el módulo a crear y es el primer archivo que deber de crear, cuando inicias con tu proyecto.

2.1.1 Explicación Manifest

A continuación explico cada punto:

- 'name': El nombre del módulo es "Teams".
- 'version': La versión del módulo es "1.0".
- 'category': La categoría del módulo es "Academica".
- 'description': Una descripción del módulo, que indica que se trata de un control de equipos.
- 'author': El autor del módulo es "Yo".
- 'maintainer': El responsable de mantener el módulo es "Yo".
- 'website': El sitio web asociado al módulo es "<http://www.itsa.edu.mx>".
- 'depends': Una lista de dependencias del módulo. En este caso, el módulo depende de "base".
- 'data': Una lista de archivos que el módulo incluye, como archivos de seguridad, vistas y archivos CSV. Estos archivos se utilizan para configurar la seguridad, las vistas y otras funcionalidades del módulo.

- 'installable': Un valor booleano que indica si el módulo es instalable. En este caso, el valor es True, lo que significa que el módulo puede ser instalado.
- 'auto_install': Un valor booleano que indica si el módulo se instala automáticamente cuando se instala una dependencia. En este caso, el valor es False, lo que significa que el módulo no se instalará automáticamente.

2.2 Modelos

En Odoo, un modelo es una representación de una entidad o concepto específico que se utiliza para almacenar y manipular datos en la base de datos. En términos más generales, un modelo en Odoo es una clase Python que define la estructura y el comportamiento de una tabla de base de datos.

Cada modelo en Odoo se corresponde con una tabla en la base de datos y está compuesto por campos que representan los diferentes atributos de la entidad que se está modelando. Estos campos pueden ser de diferentes tipos, como texto, numéricos, fechas, selecciones, entre otros.

Los modelos de Odoo también pueden contener métodos que agregan funcionalidad adicional a la entidad, como la validación de datos, cálculos personalizados, relaciones con otros modelos y acciones específicas.

Al definir un modelo en Odoo, se pueden especificar diferentes atributos, como el nombre del modelo, la descripción, los campos, los métodos, las restricciones, las relaciones con otros modelos y mucho más. Estos atributos se definen mediante código Python utilizando la API de Odoo.

Para crear un modelo en odoo es tan facil como crear un archivo con extensión .py y ahí definir tus clases de la siguiente manera:

Se define el modelo arbitros:

```
from odoo import models, fields

class arbitros(models.Model):

    _name = 'helloworld.arbitros'

    liga_id = fields.Many2one('helloworld.ligas', string='Liga')
```

```

name = fields.Char(string='Nombre')

photo = fields.Binary(string='Foto')

_order = 'liga_id,name'

```

2.2.1 Explicación Modelos

1. `from odoo import models, fields`: Esta línea importa las clases `models` y `fields` del módulo `odoo`, que son necesarias para definir el modelo y sus campos.
2. `class arbitros(models.Model)::` Esta línea define una clase Python llamada "arbitros" que hereda de la clase `models.Model` de Odoo. Esto indica que "arbitros" es un modelo de Odoo.
3. `_name = 'helloworld.arbitros'`: Esta línea define el nombre técnico del modelo, que es utilizado internamente por Odoo para identificarlo. En este caso, el nombre técnico del modelo es "helloworld.arbitros".
4. `liga_id = fields.Many2One('helloworld.ligas',string='Liga')`: Esta línea define un campo llamado "liga_id" que es de tipo `Many2One`, lo que significa que representa una relación de muchos a uno con el modelo "helloworld.ligas". El parámetro `string` establece la etiqueta o nombre descriptivo del campo que se mostrará en la interfaz de usuario.
5. `name = fields.Char(string='Nombre')`: Esta línea define un campo llamado "name" que es de tipo `Char`, lo que representa un campo de texto. El parámetro `string` establece la etiqueta o nombre descriptivo del campo.
6. `photo = fields.Binary(string='Foto')`: Esta línea define un campo llamado "photo" que es de tipo `Binary`, lo que significa que representa un archivo binario, como una imagen en este caso. El parámetro `string` establece la etiqueta o nombre descriptivo del campo.
7. `_order = 'liga_id,name'`: Esta línea establece el orden predeterminado en el que los registros del modelo "arbitros" se mostrarán en la interfaz de usuario. En este caso, los registros se ordenarán primero por el campo "liga_id" y luego por el campo "name".

El siguiente código define 2 modelos, coaches y teams

```
# -*- coding: utf-8 -*-
from odoo import models, fields

class coaches(models.Model):
    _name = 'helloworld.coachs'
    name = fields.Char(string='Nombre')
    photo = fields.Binary(string='Foto')

    _order = 'name'

class teams(models.Model):
    _name = 'helloworld.teams'

    liga_id = fields.Many2one('helloworld.ligas', string='Liga')
    name = fields.Char(string='Nombre')
    coach_id = fields.Many2one('helloworld.coachs', string='Coach')
    photo = fields.Binary(string='Foto')
    player_ids = fields.One2many('helloworld.players', 'team_id', string='Team')
    player2_ids = fields.Many2many('helloworld.players', string='Jugadores')
    playsp = fields.Integer(string='Juegos jugados')
    playsw = fields.Integer(string='Juegos ganados')
    playsl = fields.Integer(string='Juegos perdidos')
    playse = fields.Integer(string='Juegos empatados')
    _order = 'name'
```

Modelo "coachs":

- `_name = 'helloworld.coachs'`: Establece el nombre técnico del modelo como "helloworld.coachs".
- `name = fields.Char(string='Nombre')`: Define un campo llamado "name" que es de tipo Char, lo que representa un campo de texto. El parámetro string establece la etiqueta o nombre descriptivo del campo.
- `photo = fields.Binary(string='Foto')`: Define un campo llamado "photo" que es de tipo Binary, lo que representa un archivo binario, como una imagen. El parámetro string establece la etiqueta o nombre descriptivo del campo.
- `_order = 'name'`: Establece el orden predeterminado en el que los registros del modelo "coachs" se mostrarán en la interfaz de usuario. En este caso, los registros se ordenarán por el campo "name".

Modelo "teams":

- `_name = 'helloworld.teams'`: Establece el nombre técnico del modelo como "helloworld.teams".
- `liga_id = fields.Many2one('helloworld.ligas', string='Liga')`: Define un campo llamado "liga_id" que es de tipo Many2one, lo que representa una relación de muchos a uno con el modelo "helloworld.ligas". El parámetro string establece la etiqueta o nombre descriptivo del campo.

- `name = fields.Char(string='Nombre')`: Define un campo llamado "name" que es de tipo Char, lo que representa un campo de texto. El parámetro string establece la etiqueta o nombre descriptivo del campo.
- `coach_id = fields.Many2one('helloworld.coachs',string='Coach')`: Define un campo llamado "coach_id" que es de tipo Many2one, lo que representa una relación de muchos a uno con el modelo "helloworld.coachs". El parámetro string establece la etiqueta o nombre descriptivo del campo.
- `photo = fields.Binary(string='Foto')`: Define un campo llamado "photo" que es de tipo Binary, lo que representa un archivo binario, como una imagen. El parámetro string establece la etiqueta o nombre descriptivo del campo.
- `player_ids = fields.One2many('helloworld.players','team_id',string='Team')`: Define un campo llamado "player_ids" que es de tipo One2many, lo que representa una relación de uno a muchos con el modelo "helloworld.players". El parámetro string establece la etiqueta o nombre descriptivo del campo.
- `player2_ids = fields.Many2many('helloworld.players',string='Jugadores')`: Define un campo llamado "player2_ids" que es de tipo Many2many, lo que representa una relación de muchos a muchos con el modelo "helloworld.players". El parámetro string establece la etiqueta o nombre descriptivo del campo.
- `playsp, playsw, playsl, playse`: Estos campos representan el número de juegos jugados, ganados, perdidos y empatados respectivamente, y son de tipo Integer para almacenar valores numéricos enteros.

En el siguiente código se define los modelos plays y plays_arbitros:

```
# -*- coding: utf-8 -*-
from odoo import models, fields, api
from datetime import datetime
from odoo.exceptions import UserError

class plays(models.Model):
    _name = 'helloworld.plays'

    name = fields.Char(string='Folio', readonly=True)
    fecha = fields.Datetime(string='Fecha', required=True)
    liga_id = fields.Many2one('helloworld.ligas', string='Liga', required=True)
    team1_id = fields.Many2one('helloworld.teams', string='Equipo Local',
required=True)

    team2_id = fields.Many2one('helloworld.teams', string='Equipo Visitante',
required=True)

    estadio_id = fields.Many2one('helloworld.estadios', string='Estadio',
required=True)

    arbitro_ids =
fields.One2many('helloworld.plays_arbitros', 'play_id', string='Arbitros')
```

```

state =
fields.Selection([('cre','Creado'),('prog','Programado'),('rea','Realizo'),('can','
Cancelado')],string='Estado',readonly=True, default='cre')

_order = 'name'
_sql_constraints = [('plays_uniq', 'unique(name)', 'Juego duplicado, intenta con
otro...'),]

@api.model
def create(self,vals):
    vals['name'] =
self.env['ir.sequence'].next_by_code('helloworld.plays.folio')
    fecha_act = datetime.today().strftime("%Y-%m-%d %H:%M:%S")
    if vals['fecha'] < fecha_act:
        raise UserError('La fecha no debe ser menor a la actual')

    if vals['team1_id'] == vals['team2_id']:
        raise UserError('El local y visitante deben ser diferentes')

    fecha = vals['fecha']
    h = int(fecha[11:13])*3600
    m = int(fecha[14:16])*60
    s = int(fecha[17:])
    ss1 = h + m + s
    ss2 = h + m + s + 7200

    plays_prog = self.env['helloworld.plays'].search([('state','=', 'cre')])
    for play in plays_prog:
        if vals['estadio'] == play.estadio.id:
            if fecha[:10] == play.fecha.strftime("%Y-%m-%d %H:%M:%S")[:10]:
                fecha = play.fecha.strftime("%Y-%m-%d %H:%M:%S")
                h = int(fecha[11:13])*3600
                m = int(fecha[14:16])*60
                s = int(fecha[17:])
                ww1 = h + m + s
                ww2 = h + m + s + 7200
                if (ss1 >= ww1 and ss1 <= ww2) or (ss2 >= ww1 and ss2 <= ww2):
                    raise UserError('Fecha/hora empalmada')

    return super(plays,self).create(vals)

def write(self,vals):
    if vals.get('fecha'):
        fecha_act = datetime.today().strftime("%Y-%m-%d %H:%M:%S")
        if vals['fecha'] < fecha_act:
            raise UserError('La fecha no debe ser menor a la actual')

    if vals.get('team1_id'):
        equ1 = vals['team1_id']
    else:
        equ1 = self.team1_id.id

    if vals.get('team2_id'):
        equ2 = vals['team2_id']
    else:
        equ2 = self.team2_id.id

    if equ1 == equ2:
        raise UserError('El local y visitante deben ser diferentes')

    return super(plays,self).write(vals)

class plays_arbitros(models.Model):
    _name = 'helloworld.plays_arbitros'

    play_id = fields.Many2one('helloworld.plays',string='Play')

```

```

arbitro_id = fields.Many2one('helloworld.arbitros',string='Arbitro')
posicion =
fields.Selection([('central','Central'),('bandera','Bandera'),('otro','Otro')],string='Posición')

_order = 'play_id,arbitro_id'
_sql_constraints = [('plays_uniq', 'unique(play_id,arbitro_id)', 'Arbitro
duplicado, intenta con otro...'),]

```

Explicación:

Modelo "plays":

- `_name = 'helloworld.plays'`: Establece el nombre técnico del modelo como "helloworld.plays".
- `name = fields.Char(string='Folio', readonly=True)`: Define un campo llamado "name" que es de tipo Char, lo que representa un campo de texto. El parámetro string establece la etiqueta o nombre descriptivo del campo. Además, se establece `readonly=True`, lo que significa que el campo no se puede editar directamente.
- `fecha = fields.Datetime(string='Fecha', required=True)`: Define un campo llamado "fecha" que es de tipo Datetime, lo que representa una fecha y hora. El parámetro string establece la etiqueta o nombre descriptivo del campo. Además, se establece `required=True`, lo que significa que el campo es obligatorio.
- `liga_id, team1_id, team2_id, estadio_id`: Estos campos representan relaciones de muchos a uno con otros modelos. El parámetro string establece la etiqueta o nombre descriptivo del campo. Además, se establece `required=True`, lo que significa que los campos son obligatorios.
- `arbitro_ids = fields.One2many('helloworld.plays_arbitros','play_id',string='Arbitros')`: Define un campo llamado "arbitro_ids" que es de tipo One2many, lo que representa una relación de uno a muchos con el modelo "helloworld.plays_arbitros". El parámetro string establece la etiqueta o nombre descriptivo del campo.
- `state = fields.Selection([('cre','Creado'),('prog','Programado'),('rea','Realizo'),('can','Cancelado')],string='Estado',readonly=True, default='cre')`: Define un campo llamado "state" que es de tipo Selection, lo que representa una selección de opciones. El parámetro string establece la etiqueta o nombre descriptivo del campo. Además, se establece `readonly=True`, lo que significa que el campo no se puede editar directamente. El parámetro default establece el valor predeterminado del campo.
- `_order = 'name'`: Establece el orden predeterminado en el que los registros del modelo "plays" se mostrarán en la interfaz de usuario. En este caso, los registros se ordenarán por el campo "name".
- `_sql_constraints = [('plays_uniq', 'unique(name)', 'Juego duplicado, intenta con otro...')]`: Define una restricción SQL en la base de datos para asegurar que el campo "name" sea único.

En el modelo "plays" se definen dos funciones: create y write. Aquí está la descripción de cada una:

1. Función create(self, vals): Esta función se ejecuta cuando se crea un nuevo registro del modelo "plays". Sobrescribe el método create heredado de la clase Model en Odoo. Aquí hay una descripción de lo que hace esta función:
 - @api.model: Este decorador indica que la función es un método de clase y no requiere un objeto self como argumento.
 - vals: Es un diccionario que contiene los valores ingresados para crear el nuevo registro.
2. El comportamiento de la función es el siguiente:
 - Genera un número de folio único para el nuevo registro utilizando una secuencia definida en Odoo: vals['name'] = self.env['ir.sequence'].next_by_code('helloworld.plays.folio').
 - Compara la fecha del juego con la fecha actual para asegurarse de que la fecha del juego no sea menor que la fecha actual. Si es menor, se genera una excepción UserError.
 - Compara los equipos local y visitante para asegurarse de que sean diferentes. Si son iguales, se genera una excepción UserError.
 - Realiza una verificación adicional para evitar que haya juegos con fechas y horarios que se superpongan. Se consulta la base de datos para obtener todos los juegos programados y se compara la fecha y hora del nuevo juego con los juegos existentes. Si hay una superposición, se genera una excepción UserError.
 - Si todas las validaciones pasan, se llama al método create de la clase padre (super) para crear el registro.
3. Función write(self, vals): Esta función se ejecuta cuando se realiza una operación de escritura (actualización) en un registro del modelo "plays". Sobrescribe el método write heredado de la clase Model en Odoo. Aquí hay una descripción de lo que hace esta función:
 - vals: Es un diccionario que contiene los valores que se van a actualizar en el registro.
4. El comportamiento de la función es el siguiente:
 - Si se está actualizando el campo "fecha", se realiza una validación para asegurarse de que la nueva fecha no sea menor que la fecha actual. Si lo es, se genera una excepción UserError.
 - Si se está actualizando el campo "team1_id", se verifica que el equipo local y el equipo visitante sean diferentes. Si son iguales, se genera una excepción UserError.
 - Luego, se llama al método write de la clase padre (super) para realizar la operación de escritura.

Ambas funciones utilizan la clase `UserError` de Odoo para generar excepciones con mensajes personalizados que se mostrarán al usuario en caso de que se violen las validaciones establecidas.

El modelo "plays_arbitros" es más sencillo y representa la relación entre los árbitros y los juegos (plays). Contiene los siguientes campos:

- `_name = 'helloworld.plays_arbitros'`: Establece el nombre técnico del modelo como "helloworld.plays_arbitros".
- `play_id = fields.Many2one('helloworld.plays',string='Play')`: Define un campo llamado "play_id" que es de tipo `Many2one`, lo que representa una relación de muchos a uno con el modelo "helloworld.plays". El parámetro `string` establece la etiqueta o nombre descriptivo del campo.
- `arbitro_id = fields.Many2one('helloworld.arbitros',string='Arbitro')`: Define un campo llamado "arbitro_id" que es de tipo `Many2one`, lo que representa una relación de muchos a uno con el modelo "helloworld.arbitros". El parámetro `string` establece la etiqueta o nombre descriptivo del campo.
- `posicion = fields.Selection([('central','Central'),('bandera','Bandera'),('otro','Otro')],string='Posición')`: Define un campo llamado "posicion" que es de tipo `Selection`, lo que representa una selección de opciones. El parámetro `string` establece la etiqueta o nombre descriptivo del campo.

El siguiente código define el modelo `players`

```
# -*- coding: utf-8 -*-
from odoo import models, fields

class players(models.Model):
    _name = 'helloworld.players'

    team_id = fields.Many2one('helloworld.teams',string='Team')
    name = fields.Char(string='Nombre')
    photo = fields.Binary(string='Foto')
    numero = fields.Integer(string='Número de playera')
    posicion =
fields.Selection([('del','Delantero'),('def','Defensa'),('por','Portero'),('car','C
arrilero'),('cen','Central')],string='Número de playera')

    _order = 'team_id,name'
```

- `team_id = fields.Many2one('helloworld.teams',string='Team')`: Define un campo llamado "team_id" que es de tipo `Many2one`, lo que representa una relación de muchos a uno con el modelo "helloworld.teams". El parámetro `string` establece la etiqueta o nombre descriptivo del campo.
- `name = fields.Char(string='Nombre')`: Define un campo llamado "name" que es de tipo `Char`, lo que representa un campo de texto. El parámetro `string` establece la etiqueta o nombre descriptivo del campo.

- `photo = fields.Binary(string='Foto')`: Define un campo llamado "photo" que es de tipo Binary, lo que representa un archivo binario, como una imagen. El parámetro string establece la etiqueta o nombre descriptivo del campo.
- `numero = fields.Integer(string='Número de playera')`: Define un campo llamado "numero" que es de tipo Integer, lo que representa un número entero. El parámetro string establece la etiqueta o nombre descriptivo del campo.
- `posicion = fields.Selection([('del','Delantero'),('def','Defensa'),('por','Portero'),('car','Carrilero'),('cen','Central']],string='Número de playera')`: Define un campo llamado "posicion" que es de tipo Selection, lo que representa una selección de opciones. El parámetro string establece la etiqueta o nombre descriptivo del campo. Las opciones disponibles son "Delfantero", "Defensa", "Portero", "Carrilero" y "Central".

El siguiente código define los modelos estados, poblaciones y ligas:

```
# -*- coding: utf-8 -*-
from odoo import models, fields

class estados(models.Model):
    _name = 'helloworld.estados'
    name = fields.Char(string='Estado')
    _order = 'name'
    _sql_constraints = [('estados_uniq', 'unique(name)', 'Estado duplicado, intenta con otro...'),]

class poblaciones(models.Model):
    _name = 'helloworld.poblaciones'
    name = fields.Char(string='Población')
    _order = 'name'
    _sql_constraints = [('poblaciones_uniq', 'unique(name)', 'Población duplicada, intenta con otra...'),]

class ligas(models.Model):
    _name = 'helloworld.ligas'

    name = fields.Char(string='Nombre')
    photo = fields.Binary(string='Logo')
    poblacion_id = fields.Many2one('helloworld.poblaciones', string='Población')
    estado_id = fields.Many2one('helloworld.estados', string='Estado')
    teams = fields.Integer(compute='cal_num_teams', string='Equipos', readonly=True)
    team_ids = fields.One2many('helloworld.teams', 'liga_id', string='Equipos')

    _order = 'name'
    _sql_constraints = [('ligas_uniq', 'unique(name)', 'Liga duplicada, intenta con otra...'),]

    def cal_num_teams(self):
        for rec in self:
            num = 0
            for team in rec.team_ids:
                num += 1
```

```
rec.teams = num
```

Modelo "estados":

- `_name = 'helloworld.estados'`: Establece el nombre técnico del modelo como "helloworld.estados".
- `name = fields.Char(string='Estado')`: Define un campo llamado "name" que es de tipo Char, lo que representa un campo de texto. El parámetro string establece la etiqueta o nombre descriptivo del campo.
- `_order = 'name'`: Establece el orden predeterminado en el que los registros del modelo "estados" se mostrarán en la interfaz de usuario. En este caso, los registros se ordenarán alfabéticamente por el campo "name".
- `_sql_constraints = [('estados_uniq', 'unique(name)', 'Estado duplicado, intenta con otro...')]`: Define una restricción SQL para asegurarse de que los valores del campo "name" sean únicos. Si se intenta crear un registro con un valor duplicado, se generará un error con el mensaje especificado.

Modelo "poblaciones":

- `_name = 'helloworld.poblaciones'`: Establece el nombre técnico del modelo como "helloworld.poblaciones".
- `name = fields.Char(string='Población')`: Define un campo llamado "name" que es de tipo Char, lo que representa un campo de texto. El parámetro string establece la etiqueta o nombre descriptivo del campo.
- `_order = 'name'`: Establece el orden predeterminado en el que los registros del modelo "poblaciones" se mostrarán en la interfaz de usuario. En este caso, los registros se ordenarán alfabéticamente por el campo "name".
- `_sql_constraints = [('poblaciones_uniq', 'unique(name)', 'Población duplicada, intenta con otra...')]`: Define una restricción SQL para asegurarse de que los valores del campo "name" sean únicos. Si se intenta crear un registro con un valor duplicado, se generará un error con el mensaje especificado.

Modelo "ligas":

- `_name = 'helloworld.ligas'`: Establece el nombre técnico del modelo como "helloworld.ligas".
- `name = fields.Char(string='Nombre')`: Define un campo llamado "name" que es de tipo Char, lo que representa un campo de texto. El parámetro string establece la etiqueta o nombre descriptivo del campo.
- `photo = fields.Binary(string='Logo')`: Define un campo llamado "photo" que es de tipo Binary, lo que representa un archivo binario, como una imagen. El parámetro string establece la etiqueta o nombre descriptivo del campo.
- `poblacion_id = fields.Many2one('helloworld.poblaciones', string='Población')`: Define un campo llamado "poblacion_id" que es de tipo Many2one, lo que representa una relación de muchos a uno con el modelo "helloworld.poblaciones". El parámetro string establece la etiqueta o nombre descriptivo del campo.

- `estado_id = fields.Many2one('helloworld.estados',string='Estado')`: Define un campo llamado "estado_id" que es de tipo Many2one, lo que representa una relación de muchos a uno con el modelo "helloworld.estados". El parámetro string establece la etiqueta o nombre descriptivo del campo.
- `teams = fields.Integer(compute='cal_num_teams',string='Equipos',readonly=True)`: Define un campo llamado "teams" que es de tipo Integer, lo que representa un número entero. El parámetro compute especifica que el valor del campo se calculará mediante la función "cal_num_teams". El parámetro string establece la etiqueta o nombre descriptivo del campo, y el parámetro readonly lo hace de solo lectura.
- `team_ids = fields.One2many('helloworld.teams','liga_id',string='Equipos')`: Define un campo llamado "team_ids" que es de tipo One2many, lo que representa una relación de uno a muchos con el modelo "helloworld.teams". El parámetro string establece la etiqueta o nombre descriptivo del campo, y el parámetro 'liga_id' especifica el campo en el modelo relacionado que establece la relación.
- `_order = 'name'`: Establece el orden predeterminado en el que los registros del modelo "ligas" se mostrarán en la interfaz de usuario. En este caso, los registros se ordenarán alfabéticamente por el campo "name".
- `_sql_constraints = [('ligas_uniq', 'unique(name)', 'Liga duplicada, intenta con otra...')]`: Define una restricción SQL para asegurarse de que los valores del campo "name" sean únicos. Si se intenta crear un registro con un valor duplicado, se generará un error con el mensaje especificado.
- `def cal_num_teams(self)`: Define una función llamada "cal_num_teams" que se utiliza como decorador en el campo "teams". Esta función se ejecuta cada vez que se accede al campo "teams" y realiza un cálculo para determinar el número de equipos asociados a la liga.

El siguiente código define el modelo estadios:

```
# -*- coding: utf-8 -*-
from odoo import models, fields
class estadios(models.Model):
    _name = 'helloworld.estadios'

    liga_id = fields.Many2one('helloworld.ligas',string='Liga')
    name = fields.Char(string='Nombre')
    photo = fields.Binary(string='Foto')

    _order = 'liga_id,name'
```

- `_name = 'helloworld.estadios'`: Establece el nombre técnico del modelo como "helloworld.estadios".

- `liga_id = fields.Many2one('helloworld.ligas',string='Liga')`: Define un campo llamado "liga_id" que es de tipo Many2one, lo que representa una relación de muchos a uno con el modelo "helloworld.ligas". El parámetro string establece la etiqueta o nombre descriptivo del campo.
- `name = fields.Char(string='Nombre')`: Define un campo llamado "name" que es de tipo Char, lo que representa un campo de texto. El parámetro string establece la etiqueta o nombre descriptivo del campo.
- `photo = fields.Binary(string='Foto')`: Define un campo llamado "photo" que es de tipo Binary, lo que representa un archivo binario, como una imagen. El parámetro string establece la etiqueta o nombre descriptivo del campo.
- `_order = 'liga_id,name'`: Establece el orden predeterminado en el que los registros del modelo "estadios" se mostrarán en la interfaz de usuario. En este caso, los registros se ordenarán primero por el campo "liga_id" y luego por el campo "name".

El siguiente código define los modelos cedulas, cedulas_det y eventos:

```
# -*- coding: utf-8 -*-
from odoo import models, fields, api
from odoo.exceptions import UserError

class cedulas(models.Model):
    _name = 'helloworld.cedulas'

    name = fields.Char(string='Folio', readonly=True)
    play_id = fields.Many2one('helloworld.plays',string='Juego', required="True")
    play_desc = fields.Char(compute='cal_desc_play',string='Desc. Juego',
readonly=True)
    resultado = fields.Char(string='Resultado', required="True") # 1-1, 0-0, 3-2,
etc
    fecha_inicio = fields.Datetime(string='Fecha de inicio', required="True")
    fecha_fin = fields.Datetime(string='Fecha de finalización', required="True")
    player_ids =
fields.One2many('helloworld.cedula_det','cedula_id',string='Players',
required="True")
    state =
fields.Selection([('cre','Creado'),('env','Enviada'),('can','Cancelado')],string='E
stado',readonly=True, default='cre', required="True")

    _order = 'play_id'

    def registrar(self):
        #team1 =
self.env['helloworld.teams'].search([('id','=',self.play_id.team1_id.id)])
        goles = self.resultado.split('-')

        vals={}
        vals['playsp'] = self.play_id.team1_id.playsp + 1
        if goles[0] > goles[1]:
            vals['playsw'] = self.play_id.team1_id.playsw + 1
        elif goles[0] < goles[1]:
            vals['playsl'] = self.play_id.team1_id.playsl + 1
        else:
            vals['playse'] = self.play_id.team1_id.playse + 1

        self.play_id.team1_id.write(vals)

        vals={}
        vals['playsp'] = self.play_id.team2_id.playsp + 1
```

```

        if goles[1] > goles[0]:
            vals['playsw'] = self.play_id.team2_id.playsw + 1
        elif goles[1] < goles[0]:
            vals['playsl'] = self.play_id.team2_id.playsl + 1
        else:
            vals['playse'] = self.play_id.team2_id.playse + 1

        self.play_id.team2_id.write(vals)

        vals={}
        vals['state'] = 'env'
        self.write(vals)

    @api.depends('play_id')
    def cal_desc_play(self):
        for rec in self:
            if rec.play_id.team1_id.name and rec.play_id.team2_id.name:
                rec.play_desc = rec.play_id.team1_id.name + ' vs ' +
rec.play_id.team2_id.name
            else:
                rec.play_desc = 'No definido'

class cedula_det(models.Model):
    _name = 'helloworld.cedula_det'
    cedula_id = fields.Many2one('helloworld.cedulas',string='Cedula')
    player_id = fields.Many2one('helloworld.players',string='Jugador')
    eventos_ids = fields.One2many('helloworld.eventos','player_id',string='Eventos')

class eventos(models.Model):
    _name = 'helloworld.eventos'
    player_id = fields.Many2one('helloworld.cedula_det',string='Jugador')
    tipo = fields.Selection([('gol','Gol'),('tarj','Tarjeta')],string='Tipo')
    minuto = fields.Char(string='Minuto del gol') # 45, 60, 89, 100
    targeta =
fields.Selection([('amarilla','Amarilla'),('roja','Roja')],string='Targeta')

    _order = 'cedula_id,player_id'

```

Modelo "cedulas":

- `_name = 'helloworld.cedulas'`: Establece el nombre técnico del modelo como "helloworld.cedulas".
- `name = fields.Char(string='Folio', readonly=True)`: Define un campo llamado "name" que es de tipo Char, lo que representa un campo de texto. El parámetro string establece la etiqueta o nombre descriptivo del campo. El parámetro readonly se establece en True para que el campo sea de solo lectura.
- `play_id = fields.Many2one('helloworld.plays',string='Juego', required=True)`: Define un campo llamado "play_id" que es de tipo Many2one, lo que representa una relación de muchos a uno con el modelo "helloworld.plays". El parámetro string establece la etiqueta o nombre descriptivo del campo. El parámetro required se establece en True para indicar que el campo es obligatorio.
- `play_desc = fields.Char(compute='cal_desc_play',string='Desc. Juego', readonly=True)`: Define un campo llamado "play_desc" que es de tipo Char, lo que representa un campo de texto. El parámetro compute indica que el valor del campo

se calculará mediante una función. El parámetro `string` establece la etiqueta o nombre descriptivo del campo. El parámetro `readonly` se establece en `True` para que el campo sea de solo lectura.

- `resultado = fields.Char(string='Resultado', required="True")`: Define un campo llamado "resultado" que es de tipo `Char`, lo que representa un campo de texto. El parámetro `string` establece la etiqueta o nombre descriptivo del campo. El parámetro `required` se establece en `True` para indicar que el campo es obligatorio.
- `fecha_inicio = fields.Datetime(string='Fecha de inicio', required="True")`: Define un campo llamado "fecha_inicio" que es de tipo `Datetime`, lo que representa una fecha y hora. El parámetro `string` establece la etiqueta o nombre descriptivo del campo. El parámetro `required` se establece en `True` para indicar que el campo es obligatorio.
- `fecha_fin = fields.Datetime(string='Fecha de finalización', required="True")`: Define un campo llamado "fecha_fin" que es de tipo `Datetime`, lo que representa una fecha y hora. El parámetro `string` establece la etiqueta o nombre descriptivo del campo. El parámetro `required` se establece en `True` para indicar que el campo es obligatorio.
- `player_ids = fields.One2many('helloworld.cedula_det', 'cedula_id', string='Players', required="True")`: Define un campo llamado "player_ids" que es de tipo `One2many`, lo que representa una relación de uno a muchos con el modelo "helloworld.cedula_det". El parámetro 'helloworld.cedula_det' indica el modelo relacionado. El parámetro `cedula_id` establece el campo inverso en el modelo relacionado. El parámetro `string` establece la etiqueta o nombre descriptivo del campo. El parámetro `required` se establece en `True` para indicar que el campo es obligatorio.
- `state = fields.Selection([('cre', 'Creado'), ('env', 'Enviada'), ('can', 'Cancelado')], string='Estado', readonly=True, default='cre', required="True")`: Define un campo llamado "state" que es de tipo `Selection`, lo que representa una selección de opciones. El parámetro `Selection` especifica las opciones disponibles. El parámetro `string` establece la etiqueta o nombre descriptivo del campo. El parámetro `readonly` se establece en `True` para que el campo sea de solo lectura. El parámetro `default` establece el valor predeterminado del campo. El parámetro `required` se establece en `True` para indicar que el campo es obligatorio.
- `_order = 'play_id'`: Establece el orden predeterminado en el que los registros del modelo "cedulas" se mostrarán en la interfaz de usuario. En este caso, los registros se ordenarán por el campo "play_id".
- `registrar(self)`: Una función que realiza ciertas operaciones al registrar una cédula, como actualizar estadísticas y cambiar el estado.
- `cal_desc_play(self)`: Una función que calcula y establece la descripción del juego en función de los equipos involucrados.

Modelo "cedula_det":

- `_name = 'helloworld.cedula_det'`: Establece el nombre técnico del modelo como "helloworld.cedula_det".
- `cedula_id = fields.Many2one('helloworld.cedulas',string='Cedula')`: Define un campo llamado "cedula_id" que es de tipo Many2one, lo que representa una relación de muchos a uno con el modelo "helloworld.cedulas". El parámetro string establece la etiqueta o nombre descriptivo del campo.
- `player_id = fields.Many2one('helloworld.players',string='Jugador')`: Define un campo llamado "player_id" que es de tipo Many2one, lo que representa una relación de muchos a uno con el modelo "helloworld.players". El parámetro string establece la etiqueta o nombre descriptivo del campo.
- `eventos_ids = fields.One2many('helloworld.eventos','player_id',string='Eventos')`: Define un campo llamado "eventos_ids" que es de tipo One2many, lo que representa una relación de uno a muchos con el modelo "helloworld.eventos". El parámetro 'helloworld.eventos' indica el modelo relacionado. El parámetro player_id establece el campo inverso en el modelo relacionado. El parámetro string establece la etiqueta o nombre descriptivo del campo.

Modelo "eventos":

- `_name = 'helloworld.eventos'`: Establece el nombre técnico del modelo como "helloworld.eventos".
- `player_id = fields.Many2one('helloworld.cedula_det',string='Jugador')`: Define un campo llamado "player_id" que es de tipo Many2one, lo que representa una relación de muchos a uno con el modelo "helloworld.cedula_det". El parámetro string establece la etiqueta o nombre descriptivo del campo.
- `tipo = fields.Selection([('gol','Gol'),('tarj','Tarjeta')],string='Tipo')`: Define un campo llamado "tipo" que es de tipo Selection, lo que representa una selección de opciones. El parámetro Selection especifica las opciones disponibles. El parámetro string establece la etiqueta o nombre descriptivo del campo.
- `minuto = fields.Char(string='Minuto del gol')`: Define un campo llamado "minuto" que es de tipo Char, lo que representa un campo de texto. El parámetro string establece la etiqueta o nombre descriptivo del campo.
- `targeta = fields.Selection([('amarilla','Amarilla'),('roja','Roja')],string='Targeta')`: Define un campo llamado "targeta" que es de tipo Selection, lo que representa una selección de opciones. El parámetro Selection especifica las opciones disponibles. El parámetro string establece la etiqueta o nombre descriptivo del campo.
- `_order = 'cedula_id,player_id'`: Establece el orden predeterminado en el que los registros del modelo "eventos" se mostrarán en la interfaz de usuario. En este caso, los registros se ordenarán por los campos "cedula_id" y "player_id".

Estos modelos y sus campos y funciones asociadas se utilizan para representar y gestionar cédulas de juegos, detalles de cédulas y eventos relacionados en un contexto específico de la aplicación.

2.3 Init

En el archivo `__init__.py` se deben de importar todos los modelos que se han creado ya que este archivo funciona como directorio para el módulo.

```
# -*- coding: utf-8 -*-  
  
from . import teams  
  
from . import players  
  
from . import ligas  
  
from . import estadios  
  
from . import arbitros  
  
from . import plays  
  
from . import cedula  
  
from .reports import plays_report_template
```

2.4 Views

En Odoo, las vistas son una parte fundamental de la interfaz de usuario de una aplicación. Representan la forma en que los datos se presentan y se interactúa con ellos en el sistema. Las vistas definen la estructura y el diseño de las pantallas y formularios en Odoo.

Una vista en Odoo es una representación visual de uno o varios registros de una entidad, como un modelo o tabla de la base de datos. Las vistas pueden mostrar diferentes tipos de información, como campos de datos, botones, imágenes y otros elementos interactivos. Además, las vistas también pueden contener lógica y funcionalidad específica, como reglas de validación, acciones a realizar cuando se hace clic en un botón, filtros de búsqueda, entre otros.

Existen varios tipos de vistas en Odoo, entre los cuales se destacan:

1. Vista de formulario (Form View): Muestra los detalles de un registro en un formulario, donde se pueden editar los campos y realizar acciones específicas relacionadas con ese registro.
2. Vista de lista (List View): Presenta una lista de registros en forma de tabla, donde cada columna corresponde a un campo del registro. Permite realizar búsquedas, filtrar los datos y ejecutar acciones masivas sobre los registros.
3. Vista de árbol (Tree View): Similar a la vista de lista, pero con una representación jerárquica de los datos, permitiendo agrupar y mostrar datos anidados.
4. Vista de calendario (Calendar View): Muestra los registros en forma de eventos en un calendario, donde se pueden visualizar y programar actividades basadas en fechas y horas.
5. Vista de gráfico (Graph View): Presenta los datos en forma de gráficos interactivos, como gráficos de barras, gráficos circulares, gráficos de líneas, etc.

El siguiente código tiene la definición de la vista **teams_view.py**

```
<?xml version="1.0" encoding="utf-8"?>
<odoo>
  <!-- vista tipo form del modelo helloworld.teams -->
  <record model="ir.ui.view" id="teams_form">
    <field name="name">helloworld.teams.form</field>
    <field name="model">helloworld.teams</field>
    <field name="type">form</field>
    <field name="arch" type="xml">
      <form>
        <sheet>
```

```

        <group string="Datos de teams">
            <field name="liga_id"/>
            <field name="name" placeholder="Nombre del equipo" />
            <field name="coach_id" placeholder="Coach"/>
            <field name="photo" placeholder="Foto" widget="image"
options="{ 'size': [150, 150]}"/>
            <field name="player2_ids" widget="many2many_tags"/>
        </group>
        <field name="player_ids" nolabel="1" readonly="True">
            <tree editable="bottom">
                <field name="photo" widget="image"/>
                <field name="name"/>
                <field name="numero"/>
                <field name="posicion"/>
            </tree>
        </field>
    </sheet>
</form>
</field>
</record>
<!-- vista tipo tree del modelo helloworld.teams -->
<record model="ir.ui.view" id="helloworld_teams_tree">
    <field name="name">helloworld.teams.tree</field>
    <field name="model">helloworld.teams</field>
    <field name="type">tree</field>
    <field name="arch" type="xml">
        <tree string="teams" >
            <field name="liga_id"/>
            <field name="name" />
            <field name="coach_id"/>
            <field name="playsp" />
            <field name="playsw" />
            <field name="playsl" />
            <field name="playse" />
            <field name="photo" widget="image" options="{ 'size': [50,
50]}"/>
        </tree>
    </field>
</record>
<record model="ir.ui.view" id="helloworld_teams_search">
    <field name="name">helloworld.alumno.search</field>
    <field name="model">helloworld.teams</field>
    <field name="arch" type="xml">
        <search>
            <field name="name" />
            <field name="coach_id"/>
        </search>
    </field>
</record>
<!-- ACCION PARA EL MODELO helloworld.teams -->
<record id="helloworld_teams_action" model="ir.actions.act_window">
    <field name="name">Teams</field>
    <field name="res_model">helloworld.teams</field>
    <field name="view_mode">tree,form</field>
</record>
</odoo>

```

2.4.1 Explicación Views:

El código XML está utilizando la etiqueta <record> para definir diferentes elementos en Odoo, como vistas, acciones y otros. Cada <record> tiene un atributo model que indica el

tipo de elemento que se está definiendo y un atributo id que especifica un identificador único para ese elemento. A continuación, se describe cada <record> presente en el código:

1. <record model="ir.ui.view" id="teams_form">: Este <record> define una vista de formulario para el modelo "helloworld.teams". La vista tiene un nombre interno "helloworld.teams.form". La etiqueta <field> se utiliza para especificar varios atributos de la vista, como el nombre del modelo, el tipo de vista (form), y el archivo XML que contiene la estructura de la vista.
2. <record model="ir.ui.view" id="helloworld_teams_tree">: Este <record> define una vista de lista (tree view) para el modelo "helloworld.teams". La vista tiene un nombre interno "helloworld.teams.tree". Al igual que en el caso anterior, se utilizan etiquetas <field> para especificar el modelo, el tipo de vista (tree), y el archivo XML que contiene la estructura de la vista.
3. <record model="ir.ui.view" id="helloworld_teams_search">: Este <record> define una vista de búsqueda para el modelo "helloworld.teams". La vista tiene un nombre interno "helloworld.alumno.search". Al igual que en los casos anteriores, se utiliza la etiqueta <field> para especificar el modelo y el archivo XML que contiene la estructura de la vista.
4. <record id="helloworld_teams_action" model="ir.actions.act_window">: Este <record> define una acción para el modelo "helloworld.teams". La acción tiene un nombre "Teams" y el modelo de destino es "helloworld.teams". Se especifica que la vista se abrirá en los modos de vista "tree" (vista de lista) y "form" (vista de formulario).

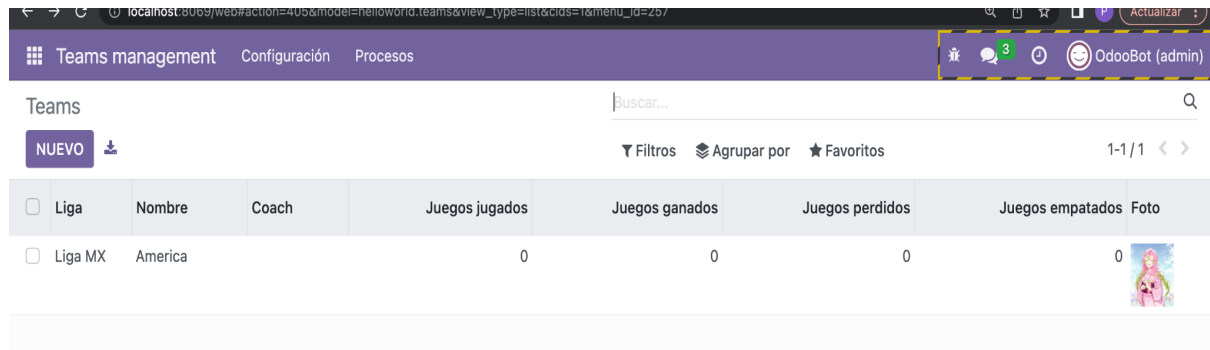
2.5 Visualización:

The screenshot displays the Odoo Teams management interface. The top navigation bar includes "Teams management", "Configuración", and "Procesos". The user is logged in as "OdooBot (admin)". The main header shows "Teams / Nuevo" and a "Nuevo" button. The form is titled "DATOS DE TEAMS" and contains the following fields:

- Liga ?**: Liga MX
- Nombre ?**: America
- Coach ?**: Mario Cocca
- Foto ?**:
- Jugadores ?**: A dropdown menu.

Below the form fields is a table with the following columns: Foto, Nombre, Núm..., and Número de playera. The table is currently empty.

Lo que se muestra en ventana es el record form el cual sirve para crear un formulario.



Este es el record tipo tree, el cual muestra los datos del formulario en una tabla, en la esquina superior derecha se encuentra una lupa la cual es un record tipo search.

El siguiente código define la vista **plays_view.py**:

```
<?xml version="1.0" encoding="utf-8"?>
<odoo>
    <!-- vista tipo form del modelo helloworld.plays -->
    <record model="ir.ui.view" id="plays_form">
        <field name="name">helloworld.plays.form</field>
        <field name="model">helloworld.plays</field>
        <field name="type">form</field>
        <field name="arch" type="xml">
            <form>
                <header>
                    <field name="state" widget="statusbar"/>
                </header>
                <sheet>
                    <group string="Datos de plays">
                        <field name="name" placeholder="Folio" />
                        <field name="fecha" placeholder="Fecha de juego"/>
                        <field name="liga_id" placeholder="Liga"/>
                        <field name="team1_id" placeholder="Liga"/>
                        <field name="team2_id" placeholder="Liga"/>
                        <field name="estadio_id" placeholder="Liga"/>
                    </group>
                    <field name="arbitro_ids" nolabel="1">
                        <tree editable="bottom">
                            <field name="arbitro_id"/>
                            <field name="posicion"/>
                        </tree>
                    </field>
                </sheet>
            </form>
        </field>
    </record>
</odoo>
```

```

        </form>
      </field>
    </record>

    <!-- vista tipo tree del modelo helloworld.plays -->
    <record model="ir.ui.view" id="helloworld_plays_tree">
      <field name="name">helloworld.plays.tree</field>
      <field name="model">helloworld.plays</field>
      <field name="type">tree</field>
      <field name="arch" type="xml">
        <tree string="plays" >
          <field name="name" />
          <field name="fecha" />
          <field name="liga_id" />
          <field name="team1_id"/>
          <field name="team2_id"/>
          <field name="estadio_id"/>
        </tree>
      </field>
    </record>

    <record model="ir.ui.view" id="helloworld_plays_search">
      <field name="name">helloworld.alumno.search</field>
      <field name="model">helloworld.plays</field>
      <field name="arch" type="xml">
        <search>
          <field name="name" />
          <field name="fecha" />
          <field name="liga_id" />
          <field name="team1_id"/>
          <field name="team2_id"/>
          <field name="estadio_id"/>
        </search>
      </field>
    </record>

    <!-- ACCION PARA EL MODELO helloworld.plays -->
    <record id="helloworld_plays_action" model="ir.actions.act_window">
      <field name="name">Plays</field>
      <field name="res_model">helloworld.plays</field>
      <field name="view_mode">tree,form</field>
    </record>

</odoo>

```

Explicación:

1. `<record model="ir.ui.view" id="plays_form">`: Este `<record>` define una vista de formulario para el modelo "helloworld.plays". La vista tiene un nombre interno "helloworld.plays.form". La etiqueta `<field>` se utiliza para especificar varios atributos de la vista, como el nombre del modelo, el tipo de vista (form), y el archivo XML que contiene la estructura de la vista. En esta vista, se definen campos como "name", "fecha", "liga_id", "team1_id", "team2_id" y "estadio_id", que se mostrarán en el formulario.
2. `<record model="ir.ui.view" id="helloworld_plays_tree">`: Este `<record>` define una vista de lista (tree view) para el modelo "helloworld.plays". La vista tiene un nombre interno "helloworld.plays.tree". Al igual que en el caso anterior, se utilizan etiquetas `<field>` para especificar el modelo, el tipo de vista (tree), y el archivo XML que

contiene la estructura de la vista. En esta vista, se definen campos como "name", "fecha", "liga_id", "team1_id", "team2_id" y "estadio_id", que se mostrarán en la lista.

3. <record model="ir.ui.view" id="helloworld_plays_search">: Este <record> define una vista de búsqueda para el modelo "helloworld.plays". La vista tiene un nombre interno "helloworld.alumno.search". Al igual que en los casos anteriores, se utiliza la etiqueta <field> para especificar el modelo y el archivo XML que contiene la estructura de la vista. En esta vista, se definen campos como "name", "fecha", "liga_id", "team1_id", "team2_id" y "estadio_id", que se utilizarán como criterios de búsqueda.
4. <record id="helloworld_plays_action" model="ir.actions.act_window">: Este <record> define una acción para el modelo "helloworld.plays". La acción tiene un nombre "Plays" y el modelo de destino es "helloworld.plays". Se especifica que la vista se abrirá en los modos de vista "tree" (vista de lista) y "form" (vista de formulario).

Visualización:

Teams management Configuración Procesos

Plays / Nuevo

Imprimir Acción Nuevo

CREADO PROGRAMADO REALIZO CANCELADO

DATOS DE PLAYS

Folio ?

Fecha ? 15/06/2023 22:59:09

Liga ? Liga MX

Equipo Local ? América

Equipo Visitante ? Chivas

Estadio ? Akron

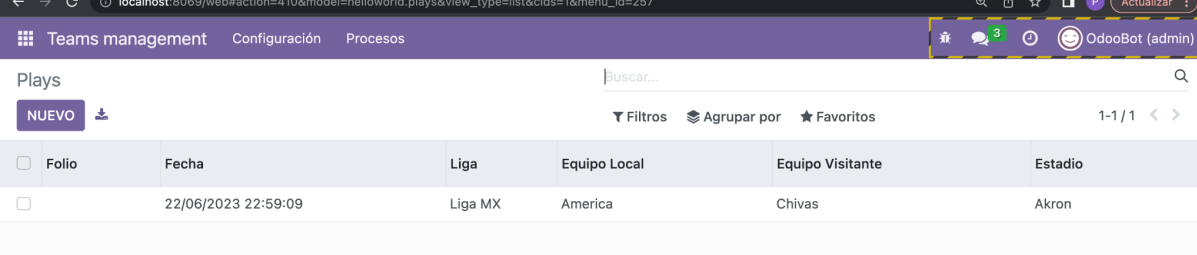
Arbitro

Posición

Agregar una línea

Campo: arbitro_ids
 Tipo: one2many
 Contexto: {}
 Dominio:
 Modificadores: {}
 Relación: helloworld.plays_arbitros

record de tipo form



The screenshot shows the Odoo Teams management interface. At the top, there's a navigation bar with 'Teams management', 'Configuración', and 'Procesos'. Below this, the 'Plays' section is active, with a search bar and a 'NUEVO' button. A table displays a single record for a play. The table has columns for 'Folio', 'Fecha', 'Liga', 'Equipo Local', 'Equipo Visitante', and 'Estadio'. The record shows a play on 22/06/2023 at 22:59:09, in Liga MX, between America and Chivas, at the Akron stadium.

Folio	Fecha	Liga	Equipo Local	Equipo Visitante	Estadio
	22/06/2023 22:59:09	Liga MX	America	Chivas	Akron

record tipo tree

El siguiente código define la vista **players_view.py**:

```
<?xml version="1.0" encoding="utf-8"?>
<odoo>
    <!-- vista tipo form del modelo helloworld.teams -->
    <record model="ir.ui.view" id="players_form">
        <field name="name">helloworld.players.form</field>
        <field name="model">helloworld.players</field>
        <field name="type">form</field>
        <field name="arch" type="xml">
            <form>
                <sheet>
                    <group string="Datos de jugadores">
                        <field name="photo" placeholder="Fotografía" widget="image"
options="{ 'size': [150, 150]}"/>
                        <field name="team_id" placeholder="Nombre del equipo" />
                        <field name="name" placeholder="Nombre del jugador"/>
                        <field name="numero" placeholder="Número de camiseta"/>
                        <field name="posicion" placeholder="Posición de juego"/>
                    </group>
                </sheet>
            </form>
        </field>
    </record>

    <!-- vista tipo tree del modelo helloworld.players -->
    <record model="ir.ui.view" id="helloworld_players_tree">
        <field name="name">helloworld.players.tree</field>
        <field name="model">helloworld.players</field>
        <field name="type">tree</field>
    </record>
</odoo>
```

```

        <field name="arch" type="xml">
            <tree string="Jugadores" >
                <field name="photo" widget="image" options="{ 'size': [50,
50]}"/>
                <field name="team_id" />
                <field name="name" />
                <field name="numero" />
                <field name="posicion" />
            </tree>
        </field>
    </record>

    <record model="ir.ui.view" id="helloworld_players_search">
        <field name="name">helloworld.players.search</field>
        <field name="model">helloworld.players</field>
        <field name="arch" type="xml">
            <search>
                <field name="team_id" />
                <field name="name"/>
                <field name="numero"/>
                <field name="posicion"/>
            </search>
        </field>
    </record>

    <record model="ir.ui.view" id="helloworld_players_kanban">
        <field name="name">helloworld.players.kanban</field>
        <field name="model">helloworld.players</field>
        <field name="arch" type="xml">
            <kanban class="o_res_partner_kanban">
                <field name="id"/>
                <field name="photo" />
                <field name="team_id" />
                <field name="name" />
                <field name="numero" />
                <field name="posicion" />
                <templates>
                    <t t-name="kanban-box">
                        <div t-attf-class="oe_kanban_global_click">
                            <div class="o_kanban_image">
                                
                            </div>
                            <div class="oe_kanban_details">
                                <strong class="o_kanban_record_title">
                                    <field name="name"/>
                                </strong>
                                <div t-if="record.numero.value">
                                    <t t-esc="record.numero.value"/>
                                </div>
                                <div t-if="record.posicion.value">
                                    <t t-esc="record.posicion.value"/>
                                </div>
                                <div t-if="record.team_id.value">
                                    <t t-esc="record.team_id.value"/>
                                </div>
                            </div>
                        </div>
                    </t>
                </templates>
            </kanban>
        </field>
    </record>

    <!-- ACCION PARA EL MODELO helloworld.players -->
    <record id="helloworld_players_action" model="ir.actions.act_window">
        <field name="name">Jugadores</field>

```

```

    <field name="res_model">helloworld.players</field>
    <field name="view_mode">kanban,form,tree</field>
  </record>
</odoo>

```

Explicación:

1. `<record model="ir.ui.view" id="players_form">`: Este `<record>` define una vista de formulario para el modelo "helloworld.players". La vista tiene un nombre interno "helloworld.players.form". Se definen campos como "photo" (fotografía), "team_id" (equipo), "name" (nombre del jugador), "numero" (número de camiseta) y "posicion" (posición de juego). La etiqueta `<field>` se utiliza para especificar varios atributos de la vista, como el nombre del modelo, el tipo de vista (form), y el archivo XML que contiene la estructura de la vista.
2. `<record model="ir.ui.view" id="helloworld_players_tree">`: Este `<record>` define una vista de lista (tree view) para el modelo "helloworld.players". La vista tiene un nombre interno "helloworld.players.tree". En esta vista, se definen campos como "photo", "team_id", "name", "numero" y "posicion" que se mostrarán en la lista de jugadores.
3. `<record model="ir.ui.view" id="helloworld_players_search">`: Este `<record>` define una vista de búsqueda para el modelo "helloworld.players". La vista tiene un nombre interno "helloworld.players.search". En esta vista, se definen campos como "team_id", "name", "numero" y "posicion" que se utilizarán como criterios de búsqueda.
4. `<record model="ir.ui.view" id="helloworld_players_kanban">`: Este `<record>` define una vista de kanban para el modelo "helloworld.players". La vista tiene un nombre interno "helloworld.players.kanban". En esta vista, se definen campos como "photo", "team_id", "name", "numero" y "posicion" que se mostrarán en las tarjetas del kanban. Además, se define una plantilla de visualización personalizada para cada tarjeta de kanban, que muestra la imagen, el nombre, el número y la posición del jugador.
5. `<record id="helloworld_players_action" model="ir.actions.act_window">`: Este `<record>` define una acción para el modelo "helloworld.players". La acción tiene un nombre "Jugadores" y el modelo de destino es "helloworld.players". Se especifica que la vista se abrirá en los modos de vista "kanban" (vista de kanban), "form" (vista de formulario) y "tree" (vista de lista).


Visualización:

← → ↻ local:localhost:8069/web?cid=1&menu_id=257&action=406&model=helloworld.players&view_type=form Actualizar

Teams management Configuración Procesos

Jugadores / Nuevo Acción Nuevo

DATOS DE JUGADORES

Foto ? 

Team ? America

Nombre ? Goku

Número de playera ? 14

Número de playera ? Delantero

record tipo form


← → ↻ local:localhost:8069/web?action=406&model=helloworld.players&view_type=kanban&cid=1&menu_id=257 Actualizar

Teams management Configuración Procesos


Jugadores Buscar...

NUEVO

▼ Filtros Agrupar por ★ Favoritos 1-2/2 < > [Iconos]



Goku
14
Delantero
America



hatsune miku
18
Delantero
America

record tipo kanban

Teams management Configuración Procesos



Actualizar

Jugadores

Buscar...

NUEVO

Filtros Agrupar por Favoritos 1-2/2

Foto	Team	Nombre	Número de playera	Número de playera
	America	Goku	14	Delantero
	America	hatsune miku	18	Delantero

record tipo tree

El siguiente código define la vista **ligas_view.py**:

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

    <!-- vista tipo tree del modelo helloworld.ligas -->
    <record model="ir.ui.view" id="helloworld_ligas_tree">
        <field name="name">helloworld.ligas.tree</field>
        <field name="model">helloworld.ligas</field>
        <field name="type">tree</field>
        <field name="arch" type="xml">
            <tree string="Ligas">
                <field name="photo" widget="image" options="{ 'size': [50,
50]}"/>
                <field name="name" />
                <field name="poblacion_id" />
                <field name="estado_id" />
                <field name="teams" />
            </tree>
        </field>
    </record>

    <record model="ir.ui.view" id="ligas_form">
        <field name="name">helloworld.ligas.form</field>
        <field name="model">helloworld.ligas</field>
        <field name="type">form</field>
        <field name="arch" type="xml">
            <form>
                <sheet>
                    <group string="Datos de ligas" >
```



```

        <field name="photo" widget="image" options="{ 'size': [150,
150]}"/>
        <field name="name" placeholder="Nombre de la liga"/>
        <field name="poblacion_id" placeholder="Población" />
        <field name="estado_id" placeholder="Estado" />
    </group>
    <field name="team_ids" nodelabel="1">
        <tree editable="bottom">
            <field name="photo" widget="image" options="{ 'size':
[50, 50]}"/>
            <field name="name"/>
            <field name="coach_id"/>
        </tree>
    </field>
</sheet>
</form>
</field>
</record>

<record model="ir.ui.view" id="helloworld_ligas_search">
    <field name="name">helloworld.ligas.search</field>
    <field name="model">helloworld.ligas</field>
    <field name="arch" type="xml">
        <search>
            <field name="name"/>
            <field name="poblacion_id"/>
            <field name="estado_id"/>
        </search>
    </field>
</record>

<!-- ACCION PARA EL MODELO helloworld.ligas -->
<record id="helloworld_ligas_action" model="ir.actions.act_window">
    <field name="name">Ligas</field>
    <field name="res_model">helloworld.ligas</field>
    <field name="view_mode">tree,form</field>
</record>

</odoo>

```

<record model="ir.ui.view" id="helloworld_ligas_tree">: Este <record> define una vista de lista (tree view) para el modelo "helloworld.ligas". La vista tiene un nombre interno "helloworld.ligas.tree". En esta vista, se definen campos como "photo" (fotografía), "name" (nombre de la liga), "poblacion_id" (población), "estado_id" (estado) y "teams" (equipos). Estos campos se mostrarán en la lista de ligas.

<record model="ir.ui.view" id="ligas_form">: Este <record> define una vista de formulario para el modelo "helloworld.ligas". La vista tiene un nombre interno "helloworld.ligas.form". Se definen campos como "photo" (fotografía), "name" (nombre de la liga), "poblacion_id" (población) y "estado_id" (estado). Además, se incluye un campo relacionado "team_ids" que se muestra como un árbol en el formulario. Los campos dentro del árbol son "photo" (fotografía del equipo), "name" (nombre del equipo) y "coach_id" (entrenador del equipo).

<record model="ir.ui.view" id="helloworld_ligas_search">: Este <record> define una vista de búsqueda para el modelo "helloworld.ligas". La vista tiene un nombre interno "helloworld.ligas.search". En esta vista, se definen campos como "name" (nombre de la liga), "poblacion_id" (población) y "estado_id" (estado) que se utilizarán como criterios de búsqueda.

<record id="helloworld_ligas_action" model="ir.actions.act_window">: Este <record> define una acción para el modelo "helloworld.ligas". La acción tiene un nombre "Ligas" y el modelo de destino es "helloworld.ligas". Se especifica que la vista se abrirá en los modos de vista "tree" (vista de lista) y "form" (vista de formulario).

Visualización:


Localhost:8069/web#id=2&cids=1&menu_id=257&action=407&model=helloworld.ligas&view_type=form

Teams management Configuración Procesos

Ligas / Liga MX

Acción 1/1 Nuevo



DATOS DE LIGAS

Logo ? 

Nombre ? Liga MX

Población ? Mexico

Estado ? CDMX

Foto	Nombre	Coach
	America	

Agregar una línea

record tipo form

Localhost:8069/web#action=407&model=helloworld.ligas&view_type=list&cids=1&menu_id=257


Teams management Configuración Procesos

Ligas

NUEVO

Buscar...

Filtros Agrupar por Favoritos 1-1/1

	Logo	Nombre	Población	Estado	Equipos
<input type="checkbox"/>		Liga MX	Mexico	CDMX	1

récord tipo tree

El siguiente código define la vista para **estadios_view.py**:

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

    <!-- vista tipo tree del modelo helloworld.estadios -->
    <record model="ir.ui.view" id="helloworld_estadios_tree">
        <field name="name">helloworld.estadios.tree</field>
        <field name="model">helloworld.estadios</field>
        <field name="type">tree</field>
        <field name="arch" type="xml">
            <tree string="Estadios" editable="bottom">
                <field name="photo" widget="image" options='{ "size": [50,
50] }' />
                <field name="name" />
                <field name="liga_id" />
            </tree>
        </field>
    </record>

    <!-- ACCION PARA EL MODELO helloworld.estadios -->
    <record id="helloworld_estadios_action" model="ir.actions.act_window">
        <field name="name">Estadios</field>
        <field name="res_model">helloworld.estadios</field>
        <field name="view_mode">tree</field>
    </record>

</odoo>
```

<record model="ir.ui.view" id="helloworld_estadios_tree">: Este <record> define una vista de lista (tree view) para el modelo "helloworld.estadios". La vista tiene un nombre interno "helloworld.estadios.tree". En esta vista, se definen campos como "photo" (fotografía del estadio), "name" (nombre del estadio) y "liga_id" (liga a la que pertenece el estadio). Estos campos se mostrarán en la lista de estadios y se podrán editar en línea (editable="bottom").

<record id="helloworld_estadios_action" model="ir.actions.act_window">: Este <record> define una acción para el modelo "helloworld.estadios". La acción tiene un nombre

"Estadios" y el modelo de destino es "helloworld.estadios". Se especifica que la vista se abrirá en el modo de vista "tree" (vista de lista).

Visualización:



Teams management Configuración Procesos

Estadios

GUARDAR DESCARTAR

Buscar...

▼ Filtros Agrupar por ★ Favoritos 1-1/1

Foto	Nombre	Liga
	Akron	Liga MX

record tipo tree

El siguiente código define la vista `cedulas_views.py`:

```
<?xml version="1.0" encoding="utf-8"?>
<odoo>
  <!-- vista tipo form del modelo helloworld.cedulas -->
  <record model="ir.ui.view" id="cedulas_form">
    <field name="name">helloworld.cedulas.form</field>
    <field name="model">helloworld.cedulas</field>
    <field name="type">form</field>
    <field name="arch" type="xml">
      <form>
        <header>
          <button name="registrar" string="Registrar" type="object"
class="oe_highlight" attrs="{ 'invisible': [ ('state', '!=', 'cre')] }"/>
          <field name="state" widget="statusbar"/>
        </header>
        <sheet>
          <group string="Datos de cedulas">
            <field name="name" placeholder="Folio" />
            <field name="fecha_inicio" placeholder="Fecha de inicio"/>
            <field name="fecha_fin" placeholder="Fecha de termino"/>
            <field name="play_id" placeholder="juego"/>
            <field name="play_desc" />
            <field name="resultado" placeholder="Resultado"/>
          </group>
          <field name="player_ids" nolabel="1">
            <tree editable="bottom">
              <field name="player_id"/>
            </tree>
          </field>
        </sheet>
      </form>
    </field>
  </record>

  <!-- vista tipo tree del modelo helloworld.cedulas -->
  <record model="ir.ui.view" id="helloworld_cedulas_tree">
    <field name="name">helloworld.cedulas.tree</field>
    <field name="model">helloworld.cedulas</field>
    <field name="type">tree</field>
    <field name="arch" type="xml">
      <tree string="cedulas">
        <field name="name" />
        <field name="fecha_inicio" />
        <field name="fecha_fin" />
        <field name="play_id"/>
        <field name="resultado"/>
        <field name="state"/>
      </tree>
    </field>
  </record>
```

```

<record model="ir.ui.view" id="helloworld_cedulas_search">
  <field name="name">helloworld.alumno.search</field>
  <field name="model">helloworld.cedulas</field>
  <field name="arch" type="xml">
    <search>
      <field name="name" />
      <field name="fecha_inicio" />
      <field name="fecha_fin" />
      <field name="play_id"/>
      <field name="resultado"/>
      <field name="state"/>
    </search>
  </field>
</record>

<!-- ACCION PARA EL MODELO helloworld.cedulas -->
<record id="helloworld_cedulas_action" model="ir.actions.act_window">
  <field name="name">Cedulas</field>
  <field name="res_model">helloworld.cedulas</field>
  <field name="view_mode">tree,form</field>
</record>
</odoo>

```

1. `<record model="ir.ui.view" id="cedulas_form">`: Este `<record>` define una vista de formulario (form view) para el modelo "helloworld.cedulas". La vista tiene un nombre interno "helloworld.cedulas.form". En esta vista, se definen campos como "name" (folio), "fecha_inicio" (fecha de inicio), "fecha_fin" (fecha de termino), "play_id" (juego), "play_desc" (descripción del juego) y "resultado" (resultado). Además, se agrega un botón llamado "Registrar" y un campo de estado que se muestra como una barra de estado.
2. `<record model="ir.ui.view" id="helloworld_cedulas_tree">`: Este `<record>` define una vista de lista (tree view) para el modelo "helloworld.cedulas". La vista tiene un nombre interno "helloworld.cedulas.tree". En esta vista, se definen campos como "name" (folio), "fecha_inicio" (fecha de inicio), "fecha_fin" (fecha de termino), "play_id" (juego), "resultado" (resultado) y "state" (estado). Estos campos se mostrarán en la lista de cedulas.
3. `<record model="ir.ui.view" id="helloworld_cedulas_search">`: Este `<record>` define una vista de búsqueda (search view) para el modelo "helloworld.cedulas". La vista tiene un nombre interno "helloworld.alumno.search". En esta vista, se definen campos como "name" (folio), "fecha_inicio" (fecha de inicio), "fecha_fin" (fecha de termino), "play_id" (juego), "resultado" (resultado) y "state" (estado). Estos campos se utilizarán para realizar búsquedas en el modelo.
4. `<record id="helloworld_cedulas_action" model="ir.actions.act_window">`: Este `<record>` define una acción para el modelo "helloworld.cedulas". La acción tiene un nombre "Cedulas" y el modelo de destino es "helloworld.cedulas". Se especifica que la vista se abrirá en los modos de vista "tree,form" (vista de lista y vista de formulario).

Visualización:

The screenshot displays the Odoo web interface for creating a new record in the 'Cedulas' model. The top navigation bar includes 'Teams management', 'Configuración', and 'Procesos'. The right sidebar shows 'Acción' and 'Nuevo' buttons. The main form area is titled 'Cedulas / Nuevo' and includes a 'REGISTRAR' button. The form fields are as follows:

DATOS DE CEDULAS	
Folio ?	
Fecha de inicio ?	22/06/2023 11:23:08
Fecha de finalización ?	23/06/2023 11:22:52
Juego ?	juego
Desc. Juego ?	No definido
Resultado ?	Resultado

A tooltip is visible over the 'Juego' field, showing the following metadata:

- Campo: player_id
- Modelo: helloworld.cedula_det
- Tipo: many2one
- Contexto: {}
- Dominio: {}
- Modificadores: {}
- Relación: helloworld.players

Below the form fields, there is a section for 'Jugador' with a dropdown menu and a button 'Agregar una línea'.

record tipo form

Por ultimo tenemos la vista para los arbitros _view.py:

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

    <!-- vista tipo tree del modelo helloworld.arbitros -->
    <record model="ir.ui.view" id="helloworld_arbitros_tree">
        <field name="name">helloworld.arbitros.tree</field>
        <field name="model">helloworld.arbitros</field>
        <field name="type">tree</field>
    </record>
</odoo>
```

```

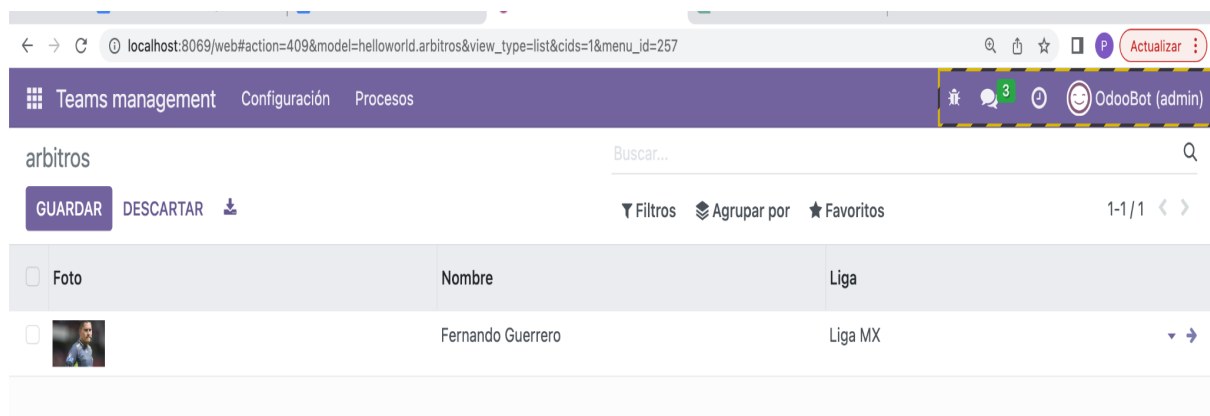
<field name="arch" type="xml" >
  <tree string="Arbitros" editable="bottom">
    <field name="photo" widget="image" options="{ 'size': [50,
50]}"/>
    <field name="name" />
    <field name="liga_id" />
  </tree>
</field>
</record>

<!-- ACCION PARA EL MODELO helloworld.arbitros -->
<record id="helloworld_arbitros_action" model="ir.actions.act_window">
  <field name="name">arbitros</field>
  <field name="res_model">helloworld.arbitros</field>
  <field name="view_mode">tree</field>
</record>

</odoo>

```

Este código sólo define una vista tipo tree



Por último tenemos que crear un menú para poder acceder a nuestras vistas creadas, eso lo hacemos igualmente en un archivo xml, pero antes de eso tenemos que hacer una configuración en el archivo manifest:

```
'data': [
```



```

    'security/security.xml',
    'security/ir.model.access.csv',
    'reports/plays_report_template.xml',
    'views/teams_view.xml',
    'views/players_view.xml',
    'views/ligas_view.xml',
    'views/estadios_view.xml',
    'views/arbitros_view.xml',
    'views/plays_view.xml',
    'views/cedulas_view.xml',
    'views/menu_view.xml',
1,

```

tenemos que agregar todas las vistas que creamos, luego creamos nuestro **menu_view.py**

```

<?xml version="1.0" encoding="utf-8"?>
<odoo>
  <data>
    <menuitem name="Teams management" id="helloworld_menu" sequence="20" />
    <menuitem name="Configuración" id="helloworld_configuracion_menu"
sequence="10" parent="helloworld_menu" />
    <menuitem name="Ligas" action="helloworld_ligas_action"
id="helloworld_ligas_menu" sequence="10" parent="helloworld_configuracion_menu" />
    <menuitem name="Estadios" action="helloworld_estadios_action"
id="helloworld_estadios_menu" sequence="20" parent="helloworld_configuracion_menu"
/>
    <menuitem name="Arbitros" action="helloworld_arbitros_action"
id="helloworld_arbitros_menu" sequence="30" parent="helloworld_configuracion_menu"
/>
    <menuitem name="Equipos" action="helloworld_teams_action"
id="helloworld_teams_menu" sequence="40" parent="helloworld_configuracion_menu" />
    <menuitem name="Procesos" id="helloworld_procesos_menu" sequence="10"
parent="helloworld_menu" />
    <menuitem name="Jugadores" action="helloworld_players_action"
id="helloworld_players_menu" sequence="50" parent="helloworld_procesos_menu" />
    <menuitem name="Plays" action="helloworld_plays_action"
id="helloworld_plays_menu" sequence="10" parent="helloworld_procesos_menu" />
    <menuitem name="Cedulas" action="helloworld_cedulas_action"
id="helloworld_cedulas_menu" sequence="20" parent="helloworld_procesos_menu" />

  </data>
</odoo>

```

3 Conclusión

En este proyecto de Odoo hemos involucrado la creación de vistas y funcionalidad para varios modelos, como ligas, estadios, cédulas y árbitros. Estas personalizaciones permiten a

los usuarios visualizar y gestionar los datos relacionados con estos elementos dentro del sistema Odoo.

Odoo, como plataforma de gestión empresarial de código abierto, ofrece una amplia gama de características y funcionalidades que se pueden adaptar a las necesidades específicas de una organización. A través de la personalización de vistas y acciones, es posible crear una experiencia de usuario más intuitiva y eficiente, alineada con los flujos de trabajo y procesos específicos de la empresa.

La flexibilidad de Odoo radica en su arquitectura modular y en la capacidad de extender y personalizar sus funcionalidades. El uso del lenguaje de marcado XML proporciona una forma estructurada y legible de definir las vistas y acciones, lo que facilita la creación y mantenimiento de personalizaciones.

4 Referencias:

[1]*Conoce Odoo 16*. (2022, 1 diciembre). Odoo S.A. Recuperado 14 de junio de 2023, de

https://www.odoo.com/es_ES/blog/odoo-news-5/conoce-odoo-16-968

[2]<https://youtu.be/3fFPGLEQTo8>