
	SISTEMAS DE GESTIÓN EMPRESARIAL	
	SGE05	
NOMBRE: Kevin Zamora Amela		
FECHA DE ENTREGA: 01/05/2025	CURSO: 2º GS	

INDICACIONES DE ENTREGA	
<p>Una vez realizada la tarea elaborarás un único documento (pdf) donde figuren las resoluciones correspondientes, y el módulo creado con todas las carpetas, y ficheros necesarios para comprobar su correcto funcionamiento, en caso contrario la actividad será evaluada como 0. El envío se realizará a través de la plataforma de la forma establecida para ello, y el archivo se nombrará siguiendo las siguientes pautas:</p> <p>apellido1_apellido2_nombre_SIGxx_Tarea.pdf</p> <p>Asegúrate que el nombre no contenga la letra ñ, tildes ni caracteres especiales extraños. Así por ejemplo la alumna Begoña Sánchez Mañas para la segunda unidad del MP de SGE, debería nombrar esta tarea como...</p> <p>sanchez_manas_begona_SGE05_Tarea.pdf</p>	
CONSEJOS Y RECOMENDACIONES	
Para realizar los ejercicios puedes consultar los tutoriales de la unidad	
RECURSOS NECESARIOS PARA REALIZAR LA TAREA	
<ul style="list-style-type: none"> • Ordenador personal. • Sistema operativo Windows o Linux. • VirtualBox o Docker • Odoo • Visual Studio Code 	
5. Desarrolla componentes para un sistema ERP-CRM analizando y utilizando el lenguaje de programación incorporado	Calificación Total
CE a) Se han reconocido las sentencias del lenguaje propio del sistema ERP-CRM +	+0,5
CE b) Se han utilizado los elementos de programación del lenguaje para crear componentes de manipulación de datos.	+0,5
CE c) Se han modificado componentes software para añadir nuevas funcionalidades al sistema.	+0,5
CE d) Se han integrado los nuevos componentes software en el sistema ERP-CRM.	+2
CE e) Se ha verificado el correcto funcionamiento de los componentes creados.	+3
CE f) Se han documentado todos los componentes creados o modificados.	+1

Enunciado

En esta unidad hemos aprendido cómo crear nuevos componentes utilizando sentencias del lenguaje propio del sistema ERP-CRM. Hemos creado componentes de manipulación de datos mediante módulos que crean a su vez tablas en la base de datos. También hemos añadido módulos al sistema y comprobado que funcionan. Además, hemos conocido herramientas adicionales para la creación de formularios e informes.

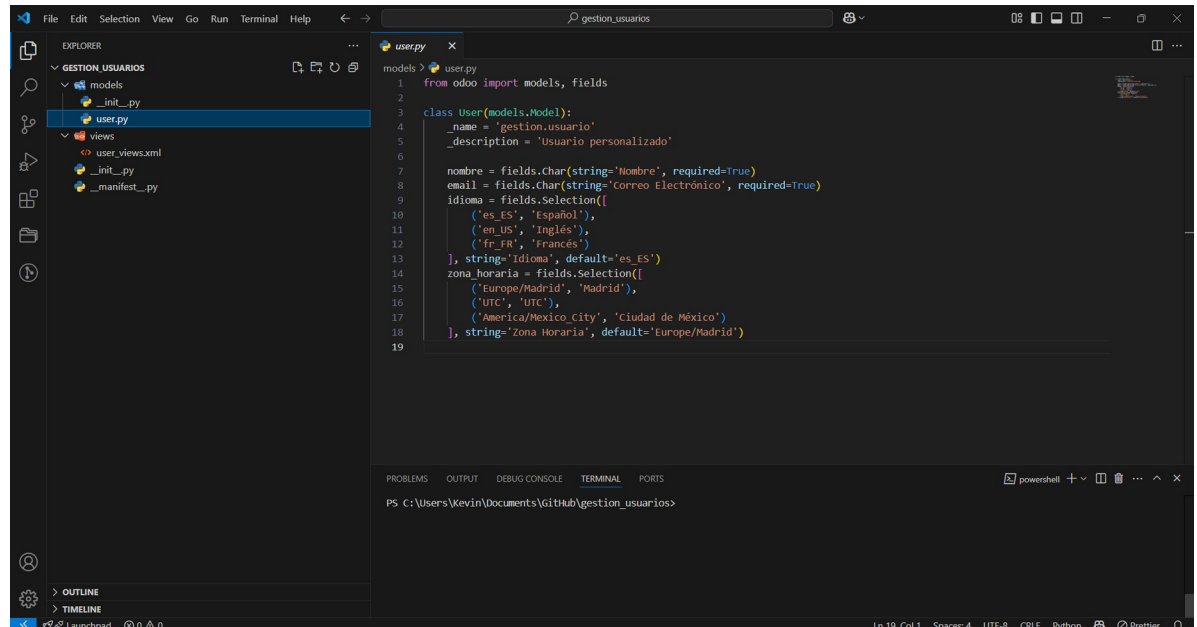
Ahora es el momento de poner en práctica estos conocimientos. La tarea consiste en crear un componente o módulo que gestione la información de usuarios con las siguientes características.

Modelo.

- Objeto en la aplicación llamado user
- Vista.
- Menú en la aplicación que enlace al objeto.
- Vista formulario con los datos del objeto. (campos: nombre, email, idioma y zona horaria)
- Vista árbol con los mismos datos del objeto. (campos: nombre, email, idioma y zona horaria).

Además del módulo deberás escribir también un informe con todas las consideraciones oportunas que se necesiten para entender cómo has realizado la tarea.

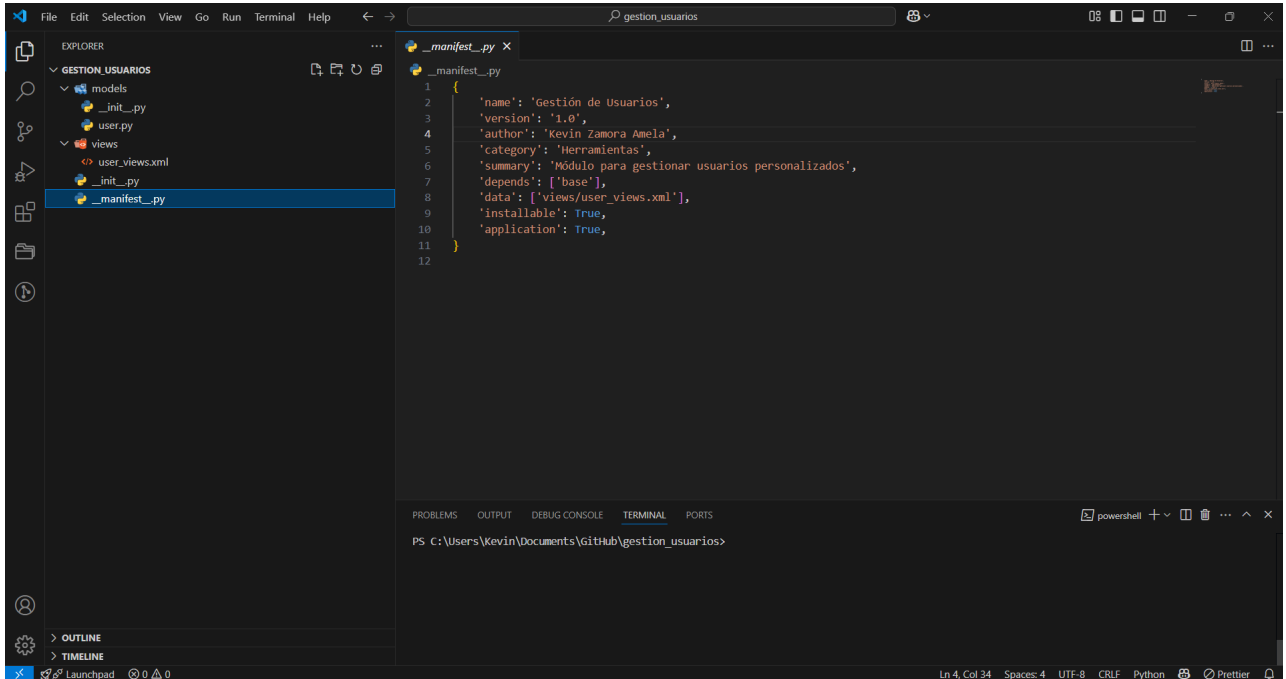
- Creamos un directorio vacío e inicializamos el componente creando a su vez todos sus archivos/componentes principales, siguiendo nuestro conocimiento previo y también las instrucciones presentadas por el enunciado anterior.



Y una vez hemos creado la estructura de nuestro componente, empezamos a introducir el código pertinente en cada archivo.

- En los archivos “__init__.py” del directorio raíz introduciremos las importaciones que nos permitirán dirigirnos y hacer accesibles la carpeta “models” y la entidad/clase “user.py” que se encuentra alojada en su interior.
- Para implementar la clase “user” introducimos el fragmento de código que podemos apreciar en la imagen adjunta que se muestra un poco más arriba. En la citada captura de pantalla, se muestra: el “IDE” utilizado, la estructura de nuestro componente y también el código “de clase” ya mencionado. Dicho código contiene principalmente la declaración de la clase (en lenguaje Python), la definición de los parámetros de propio componente y la definición de los diferentes parámetros requeridos: nombre, email, idioma y zona horaria.
- El archivo “__Manifest__.py” contiene la información/“metadatos” que identifican y representan a nuestro componente.

- Y el archivo “user_views.xml” contiene la estructura del “layout/diseño” para “construir/importar/representar” nuestro componente desde la interfaz gráfica de Odoo.
- __manifest__.py:



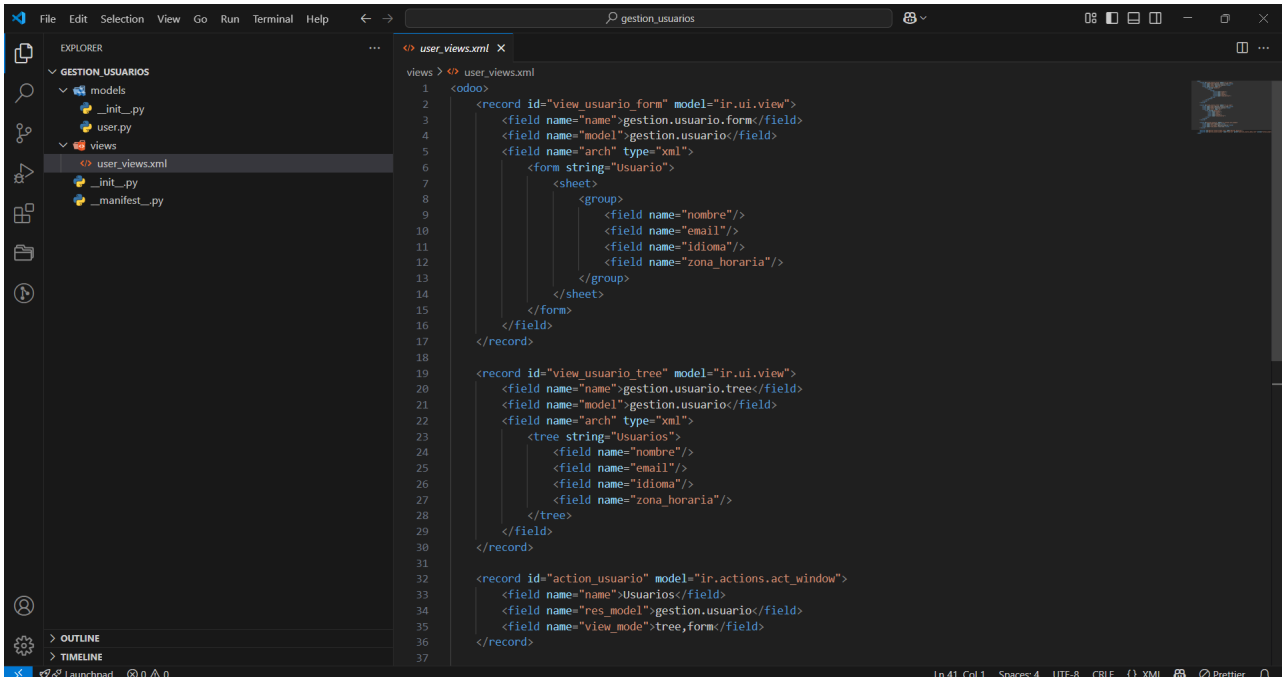
The screenshot shows the Visual Studio Code editor with the file explorer on the left displaying the project structure for 'GESTION_USUARIOS'. The file '__manifest__.py' is selected. The main editor window shows the content of this file, which is a Python dictionary defining the module's metadata. The terminal at the bottom shows the command prompt 'PS C:\Users\Kevin\Documents\GitHub\gestion_usuarios>'.

```

1  {
2      'name': 'Gestión de Usuarios',
3      'version': '1.0',
4      'author': 'Kevin Zamora Amela',
5      'category': 'Herramientas',
6      'summary': 'Módulo para gestionar usuarios personalizados',
7      'depends': ['base'],
8      'data': ['views/user_views.xml'],
9      'installable': True,
10     'application': True,
11 }
12

```

- user_views.xml:



The screenshot shows the Visual Studio Code editor with the file explorer on the left displaying the project structure for 'GESTION_USUARIOS'. The file 'user_views.xml' is selected. The main editor window shows the content of this file, which is an XML document defining the user interface views. The terminal at the bottom shows the command prompt 'PS C:\Users\Kevin\Documents\GitHub\gestion_usuarios>'.

```

1  <?xml version="1.0" encoding="utf-8"?>
2  <odoo>
3      <record id="view_usuario_form" model="ir.ui.view">
4          <field name="name">gestion.usuario.form</field>
5          <field name="model">gestion.usuario</field>
6          <field name="arch" type="xml">
7              <form string="Usuario">
8                  <sheet>
9                      <group>
10                         <field name="nombre"/>
11                         <field name="email"/>
12                         <field name="idioma"/>
13                         <field name="zona_horaria"/>
14                     </group>
15                 </sheet>
16             </form>
17         </field>
18     </record>
19     <record id="view_usuario_tree" model="ir.ui.view">
20         <field name="name">gestion.usuario.tree</field>
21         <field name="model">gestion.usuario</field>
22         <field name="arch" type="xml">
23             <tree string="Usuarios">
24                 <field name="nombre"/>
25                 <field name="email"/>
26                 <field name="idioma"/>
27                 <field name="zona_horaria"/>
28             </tree>
29         </field>
30     </record>
31     <record id="action_usuario" model="ir.actions.act_window">
32         <field name="name">Usuarios</field>
33         <field name="res_model">gestion.usuario</field>
34         <field name="view_mode">tree,form</field>
35     </record>
36 </odoo>
37

```

Error al intentar importar el módulo creado (utilizando diferentes ubicaciones de partida)

