

TECNOLÓGICO NACIONAL DE MÉXICO



INSTITUTO TECNOLÓGICO SUPERIOR
DE APATZINGÁN

Reporte de Practica



Materia: Taller de Programación Web II

Unidad: 2

Docente: I.S.C Javier Cisneros Lucatero

Alumnos: Pedro Cabello Mondragón 1902085

Eduardo Armenta Lopez 19020296

Ramon Antonio Tarelo Cerna 19020365

Felix Enrique Chavez Navarro 19020270

Actividad: Realizar Módulo de Oddo 16

Fecha de entrega: 26/06/2023

índice:

1 Introducción:	3
2 _manifest_.py:	3
3 Backend:	4
3.1 habitaciones.py	5
3.2 huespedes.py	6
3.3 reservaciones.py	7
4 _init_.py	9
5 Frontend:	10
5.1 habitaciones_view.xml	10
5.2 huespedes_view.xml	12
5.3 menu_view.xml	14
5.4 reserrvaciones_view.xml	15
6 Funcionamiento	17
7 Conclusión:	19

1 Introducción:

El siguiente proyecto nos proporciona una descripción detallada de la función del código relacionado con el proyecto de gestión del hotel en el entorno de Odoo. Se explora una serie de aspectos clave, como la importación de módulos, la estructura del manifiesto del módulo, la definición de modelos de datos y se muestra una página funcional del hotel.comenzaremos explicando el proceso de importación de módulos, donde se menciona que se importan los módulos de habitaciones, huéspedes y reservaciones desde el mismo paquete que el archivo actual. Se destaca que el punto antes del nombre del módulo indica que se está importando desde el mismo paquete. Esta información sienta las bases para comprender la estructura modular del proyecto. Y luego, se profundiza en el manifiesto del módulo, el cual define datos esenciales sobre el mismo. Se enumeran y explican las claves presentes en el manifiesto. Esta sección proporciona una comprensión integral de la información clave asociada al módulo. en si el propósito de desarrollo de esta práctica es poder conocer y comprender sobre el uso de la herramienta odoo mediante el desarrollo de módulos y estructurando el código de tal manera que podamos trabajar con los métodos y sintaxis de python el manejo de esta práctica de hotel ayudará también a poder embarcar futuros proyectos aún más grandes como un hotel 5 estrellas.

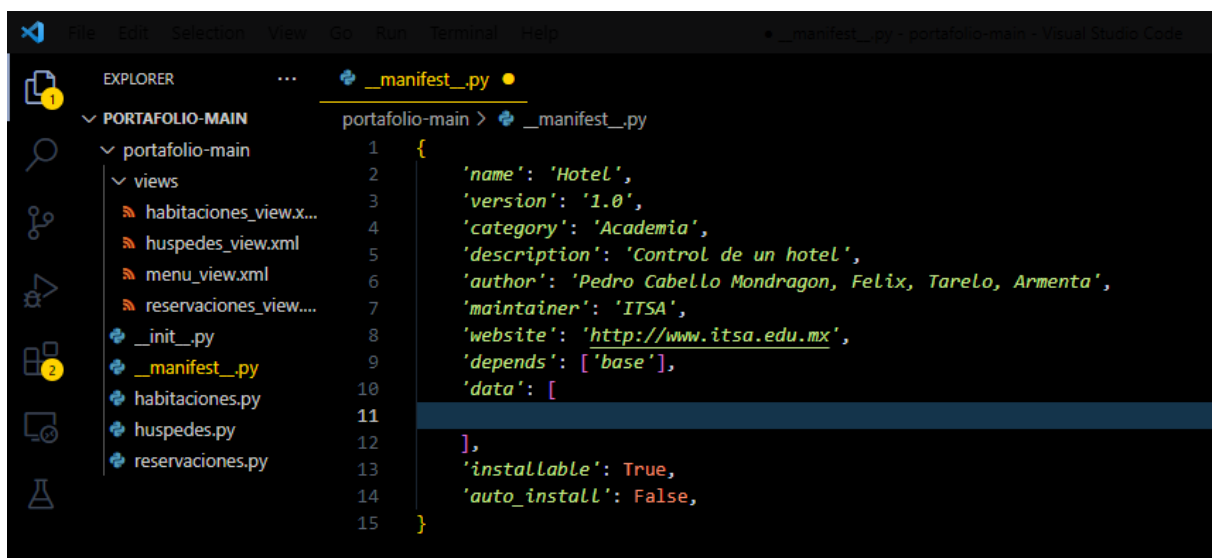
2 `_manifest_.py`:

Este código de manifiesto del módulo de Odoo. El manifiesto define varios metadatos sobre el módulo, como su nombre, versión, categoría, descripción, autor, sitio web, dependencias y otros detalles.

a continuación se explica cada una de las claves en el manifest:

- 'name': El nombre del módulo.
- 'version': La versión del módulo.
- 'category': La categoría del módulo. En este caso, parece ser "Academia".
- 'description': Una descripción del módulo.
- 'author': El autor o autores del módulo.
- 'maintainer': El mantenimiento del módulo.
- 'website': El sitio web del autor o del proyecto.
- 'depends': Las dependencias del módulo. En este caso, parece que el módulo depende del módulo "base".
- 'data': Una lista de archivos XML que definen la interfaz de usuario del módulo.
- 'installable': Un indicador booleano que especifica si el módulo se puede instalar o no.
- 'auto_install': Un indicador booleano que especifica si el módulo se instalará automáticamente como una dependencia de otro módulo.

En general, este archivo de manifiesto se utiliza para describir el módulo y sus características, así como para especificar su comportamiento de instalación.



```

1  {
2      'name': 'Hotel',
3      'version': '1.0',
4      'category': 'Academia',
5      'description': 'Control de un hotel',
6      'author': 'Pedro Cabello Mondragon, Felix, Tarelo, Armenta',
7      'maintainer': 'ITSA',
8      'website': 'http://www.itsa.edu.mx',
9      'depends': ['base'],
10     'data': [
11
12     ],
13     'installable': True,
14     'auto_install': False,
15 }
  
```

3 Backend:

en esta parte todo lo relacionado con la lógica del proyecto en este caso se trata de un hotel por lo que se llevara acabo la parte logica que interactua con el servidor por asi decirlo se encarga de todas las operaciones y procesamiento de datos detrás de escena, permitiendo que la aplicación funcione correctamente y brinde todas sus funcionalidades a los usuarios.

3.1 habitaciones.py

En este código define una clase Python llamada "HotelRoom" que hereda de "models.Model" de Odoo, lo que indica que esta clase es un modelo de datos en Odoo.

La clase tiene cuatro campos:

- "name": un campo Char que representa el número de habitación.
- "tipo_de_habitacion": un campo Selection que permite elegir entre tres opciones de tipo de habitación: "Single", "Double" o "Suite".
- "precio_por_noche": un campo Float que representa el precio de la habitación por noche.
- "estado": un campo Selection que indica si la habitación está disponible, ocupada o en mantenimiento.

El método "@api.onchange" es un decorador que se utiliza para especificar una función que se debe ejecutar automáticamente cuando se cambia el valor de un campo. En este caso, la función "onchange_tipo_de_habitacion" se ejecuta cuando se cambia el valor del campo "tipo_de_habitacion". Esta función ajusta automáticamente el valor del campo "precio_por_noche" según el tipo de habitación seleccionado.

```
portafolio-main > habitaciones.py > HotelRoom
1  from odoo import models, fields, api
2
3  class HotelRoom(models.Model):
4      _name = 'hotel.room'
5      _description = 'Hotel Room'
6
7      name = fields.Char(string='Numero de habitación')
8      tipo_de_habitacion = fields.Selection([
9          ('single', 'Single'),
10         ('double', 'Double'),
11         ('suite', 'Suite')
12     ], string='Tipo de Habitación', required=True)
13     precio_por_noche = fields.Float(string='Precio por Noche', required=True)
14     estado = fields.Selection([
15         ('available', 'Disponible'),
16         ('occupied', 'Ocupada'),
17         ('maintenance', 'Mantenimiento')
18     ], string='Estado', default='available')
```

El método "@api.onchange" es un decorador que se utiliza para especificar una función que se debe ejecutar automáticamente cuando se cambia el valor de un campo. En este caso, la función "onchange_tipo_de_habitacion" se ejecuta

cuando se cambia el valor del campo "tipo_de_habitacion". Esta función ajusta automáticamente el valor del campo "precio_por_noche" según el tipo de habitación seleccionado.

```

19
20     @api.onchange('tipo_de_habitacion')
21     def onchange_tipo_de_habitacion(self):
22         if self.tipo_de_habitacion == 'single':
23             self.precio_por_noche = 400.0
24         elif self.tipo_de_habitacion == 'double':
25             self.precio_por_noche = 650.0
26         elif self.tipo_de_habitacion == 'suite':
27             self.precio_por_noche = 120.0
28

```

Además, hay otro método llamado "onchange_habilitar" que no está siendo utilizado. Este método establece el campo "estado" en "available", pero no se especifica cuándo o cómo se debe llamar a esta función.

```

29
30     def onchange_habilitar(self):
31         self.estado = 'available'
32

```

3.2 huespedes.py

El siguiente código define una clase Python llamada "HotelGuest" que también hereda de "models.Model" de Odoo, lo que indica que esta clase también es un modelo de datos en Odoo.

La clase tiene seis campos:

- "name": un campo Char que representa el nombre del huésped.
- "last_name": un campo Char que representa el apellido del huésped.
- "phone_number": un campo Char que representa el número de teléfono del huésped.
- "email": un campo Char que representa el correo electrónico del huésped.
- "check_in_date": un campo Date que representa la fecha de ingreso del huésped al hotel.
- "check_out_date": un campo Date que representa la fecha de salida del huésped del hotel.

Estos campos se utilizan para almacenar información sobre los huéspedes del hotel, como su nombre, apellido, información de contacto y fechas de check-in y check-out.

```
portafolio-main > huspedes.py > ...
1  from odoo import models, fields
2
3  class HotelGuest(models.Model):
4      _name = 'hotel.guest'
5      _description = 'Hotel Guest'
6
7      name = fields.Char(string='Nombre')
8      last_name = fields.Char(string='Apellido')
9      phone_number = fields.Char(string='Número de Teléfono')
10     email = fields.Char(string='Correo Electrónico')
11     check_in_date = fields.Date(string='Fecha de Ingreso')
12     check_out_date = fields.Date(string='Fecha de Salida')
13
```

3.3 reservaciones.py

Este código es una definición del modelo de datos HotelReservation en Odoo. La clase HotelReservation hereda de models.Model de Odoo, lo que significa que esta clase es un modelo de datos que representa una tabla de la base de datos.

El atributo `_name` define el nombre del modelo en la base de datos. En este caso, el modelo se llama `hotel.reservation`. El atributo `_description` es una cadena que describe el modelo.

Los campos del modelo se definen como atributos en la clase, utilizando las clases `fields` de Odoo. Los campos incluyen `reservation_id`, `guest_id`, `room_id`, `precio_por_noche`, `check_in_date`, `noches`, `check_out_date`, `status` y `precio_total`.

Los campos `reservation_id`, `guest_id` y `room_id` son campos relacionales que se refieren a los modelos `hotel.reservation`, `hotel.guest` y `hotel.room`, respectivamente.

El campo `precio_por_noche` es un campo calculado que obtiene su valor del campo `precio_por_noche` del modelo `hotel.room`.

```

portafolio-main > reservaciones.py > HotelReservation
1  from odoo import models, fields, api
2
3  class HotelReservation(models.Model):
4      _name = 'hotel.reservation'
5      _description = 'Hotel Reservation'
6
7      reservation_id = fields.Many2one('hotel.reservation')
8      guest_id = fields.Many2one('hotel.guest', string='Huésped')
9      room_id = fields.Many2one('hotel.room', string='Habitación')
10     precio_por_noche = fields.Float(string='Precio por noche', compute='_compute_price_per_night')
11     check_in_date = fields.Date(string='Fecha de Ingreso')
12     noches = fields.Char(string='Numero de noches')
13     check_out_date = fields.Date(string='Fecha de Salida')
14     status = fields.Selection([
15         ('available', 'Disponible'),
16         ('occupied', 'Ocupada'),
17         ('maintenance', 'Mantenimiento')
18     ], string='Estado', default='available', compute='_compute_status')
19     precio_total = fields.Float(string='Precio Total')
20
21

```

El método `onchange_room_id` se ejecuta cuando se cambia la habitación en una reserva y actualiza los campos de la reserva y los huéspedes correspondientes. También calcula el número de noches y el precio total de la reserva.

```

22     #cambiar las fechas para el huésped desde otro evento
23     def onchange_room_id(self):
24         if self.room_id:
25             self.room_id.write({'estado': 'occupied'})
26             self.status = 'occupied'
27         if self.guest_id:
28             self.guest_id.write({'check_in_date': self.check_in_date})
29             self.guest_id.write({'check_out_date': self.check_out_date})
30             nights = (self.check_out_date - self.check_in_date).days
31             self.write({'noches': nights})
32             self.precio_total = self.room_id.precio_por_noche * nights
33
34

```

El método `onchange_desocupar` se ejecuta cuando se desocupa una habitación y borra el registro correspondiente de la base de datos.

```

34
35     def onchange_desocupar(self):
36         if self.room_id:
37             self.room_id.write({'estado': 'maintenance'})
38             self.status = 'maintenance'
39         if self.guest_id:
40             self.guest_id.unlink()
41             self.guest_id.write({'active': False})
42         self.reservation_id.unlink()
43         self.reservation_id.write({'active': False})
44
45

```


Los métodos `_compute_price_per_night` y `_compute_status` son decoradores de dependencia que se ejecutan cuando se accede a los campos correspondientes.

```
45     @api.depends('room_id')
46     def _compute_price_per_night(self):
47         for reservation in self:
48             reservation.precio_por_noche = reservation.room_id.precio_por_noche
49
```

Dentro del método `_compute_status`, se itera a través de cada instancia de la reserva (representada por `self`). La variable `reservation` representa cada instancia individual en cada iteración.

La línea `reservation.status = reservation.room_id.estado` asigna el valor del campo `estado` del objeto `room_id` (que es una relación a otro modelo llamado `room`) al campo `status` de la reserva actual. Esto implica que el campo `status` de la reserva se actualiza con el valor del campo `estado` del objeto `room_id` relacionado.

```
50     @api.depends('room_id')
51     def _compute_status(self):
52         for reservation in self:
53             reservation.status = reservation.room_id.estado
```

4 __init__.py

Este código importará módulos (habitaciones, huéspedes y reservaciones) desde el mismo paquete que el archivo actual. El punto antes del nombre del módulo indica que se está importando desde el mismo paquete que el archivo actual.



```

__manifest__.py  __init__.py X
portafolio-main > __init__.py
1  from . import habitaciones
2  from . import huspedes
3  from . import reservaciones
4
5

```

5 Frontend:

el frontend de esta aplicación de Odoo es la parte visible de la aplicación con la que los usuarios interactúan, y está basado en tecnologías web como HTML, CSS y JavaScript, pero en este caso nuestro frontend los crearemos con archivos xml junto con el uso de frameworks como Vue.js para crear una interfaz de usuario rica y dinámica.

5.1 habitaciones_view.xml

view_hotel_room_form: es una vista que define la apariencia y funcionalidad de la vista de formulario de un registro de la clase hotel.room. En el archivo XML, se especifica que el formulario tendrá una pestaña (sheet) con un encabezado (header) que contiene un widget de barra de estado (statusbar) basado en el campo estado de la clase hotel.room. En el cuerpo del formulario (group), hay tres campos: tipo_de_habitacion, name y precio_por_noche. Además, hay un botón llamado "Habilitar", que llama al método onchange_habilitar de la clase hotel.room cuando se hace clic en él.

```

portafolio-main > views > habitaciones_view.xml
1  <?xml version="1.0" encoding="utf-8"?>
2  <odoo>
3      <record id="view_hotel_room_form" model="ir.ui.view">
4          <field name="name">hotel.room.form</field>
5          <field name="model">hotel.room</field>
6          <field name="arch" type="xml">
7              <form string="Habitación">
8                  <sheet>
9                      <header>
10                         <field name="estado" widget="statusbar"/>
11                     </header>
12                     <group>
13                         <field name="tipo_de_habitacion" on_change="onchange_tipo_de_habitacion(tipo_de_habitacion, precio_
14                         <field name="name"/>
15                         <field name="precio_por_noche"/>
16                         <button name="onchange_habilitar" string="Habilitar" type="object" class="btn-primary" attrs='{ 'in
17                     </group>
18                 </sheet>
19             </form>
20         </field>
21     </record>

```

view_hotel_room_tree: es una vista que define la apariencia y funcionalidad de la vista de lista (vista en árbol) de todos los registros de la clase hotel.room. En el archivo XML, se especifica que la vista tendrá cuatro columnas: tipo_de_habitacion, name, precio_por_noche y estado.

```

22
23  ✓   <record id="view_hotel_room_tree" model="ir.ui.view">
24      <field name="name">hotel.room.tree</field>
25      <field name="model">hotel.room</field>
26      <field name="arch" type="xml">
27          <tree string="Habitaciones">
28              <field name="tipo_de_habitacion"/>
29              <field name="name"/>
30              <field name="precio_por_noche"/>
31              <field name="estado"/>
32          </tree>
33      </field>
34  </record>

```

action_hotel_rooms: es una acción que define cómo se debe abrir la vista de lista de todos los registros de la clase hotel.room. En el archivo XML, se especifica que la acción se llamará "Habitaciones", la clase del modelo asociado es hotel.room, y se abrirá en modo de vista en árbol (tree) y formulario (form).

```
35
36  ✓  <record id="action_hotel_rooms" model="ir.actions.act_window">
37      <field name="name">Habitaciones</field>
38      <field name="res_model">hotel.room</field>
39      <field name="view_mode">tree,form</field>
40  </record>
41
42  </odoo>
```

5.2 huespedes_view.xml

Este es un fragmento de código XML utilizado en la definición de vistas y acciones de la aplicación de gestión de hoteles en Odoo.

El código define tres registros utilizando la etiqueta `record`. Cada registro se refiere a una vista o acción específica de la aplicación.

El primer registro define una vista de formulario (form view) para el modelo `hotel.guest`. La vista de formulario se define dentro de la etiqueta `arch` y contiene una sola hoja (sheet) que agrupa un conjunto de campos (field) que se muestran en la vista.

```

ortafolio-main > views > huspedes_view.xml
1  <?xml version="1.0" encoding="utf-8"?>
2  <odoo>
3      <record id="view_hotel_guest_form" model="ir.ui.view">
4          <field name="name">hotel.guest.form</field>
5          <field name="model">hotel.guest</field>
6          <field name="arch" type="xml">
7              <form string="Huésped">
8                  <sheet>
9                      <group>
10                         <field name="name"/>
11                         <field name="last_name"/>
12                         <field name="phone_number"/>
13                         <field name="email"/>
14                         <field name="check_in_date" readonly="1"/>
15                         <field name="check_out_date" readonly="1"/>
16                     </group>
17                 </sheet>
18             </form>
19         </field>
20     </record>

```

El segundo registro define una vista de lista (tree view) para el mismo modelo hotel.guest. La vista de lista se define de manera similar a la vista de formulario, pero en lugar de una hoja, se utiliza una etiqueta tree.

```

21
22     <record id="view_hotel_guest_tree" model="ir.ui.view">
23         <field name="name">hotel.guest.tree</field>
24         <field name="model">hotel.guest</field>
25         <field name="arch" type="xml">
26             <tree string="Huéspedes">
27                 <field name="name"/>
28                 <field name="last_name"/>
29                 <field name="phone_number"/>
30                 <field name="email"/>
31                 <field name="check_in_date"/>
32                 <field name="check_out_date"/>
33             </tree>
34         </field>
35     </record>

```

El tercer registro define una acción de ventana (act_window) para el modelo hotel.guest. La acción de ventana se utiliza para abrir una ventana emergente que muestra la lista o formulario de registros del modelo. Esta acción especifica el

nombre de la acción, el modelo que se está abriendo y los modos de vista (view_mode) que se usarán para mostrar los registros.

```

37 <record id="action_hotel_guests" model="ir.actions.act_window">
38   <field name="name">Huéspedes</field>
39   <field name="res_model">hotel.guest</field>
40   <field name="view_mode">tree,form</field>
41 </record>
42 </odoo>

```

5.3 menu_view.xml

Este código define una estructura de menú en el módulo de Odoo. Los elementos menuitem representan las opciones de menú que estarán disponibles en la interfaz de usuario.

El primer elemento menuitem con el atributo id "hotel_menu" es el menú principal "Hotel". A continuación, se define un submenú "Administración" con el atributo id "hotel_configuracion_menu" y se asigna al menú principal "Hotel" mediante el atributo parent.

Luego se definen tres elementos menuitem más para los submenús de "Habitaciones", "Huspedes" y "Reservaciones" respectivamente. Cada uno tiene un action que se vincula con la funcionalidad del módulo.

Finalmente, todos los elementos menuitem están contenidos dentro del elemento data que es un requisito del formato XML utilizado por Odoo.

```

portfolio-main > views > menu_view.xml
1 <?xml version="1.0" encoding="utf-8"?>
2 <odoo>
3   <data>
4     <menuitem name="Hotel" id="hotel_menu" sequence="20" />
5     <menuitem name="Administracion" id="hotel_configuracion_menu" sequence="10" parent="hotel_menu" />
6     <menuitem name="Habitaciones" action="action_hotel_rooms" id="hotel_habitaciones_menu" sequence="20"
7       parent="hotel_configuracion_menu" />
8     <menuitem name="Huspedes" action="action_hotel_guests" id="hotel_huspedes_menu" sequence="30"
9       parent="hotel_configuracion_menu" />
10    <menuitem name="Reservaciones" action="action_hotel_reservations" id="hotel_reservaciones_menu"
11      sequence="40" parent="hotel_configuracion_menu" />
12  </data>
13 </odoo>

```

5.4 reservaciones_view.xml

Este código XML define las vistas y acciones relacionadas con el modelo "hotel.reservation" en Odoo.

La primera parte del código define una vista de formulario con id "view_hotel_reservation_form". Esta vista muestra un formulario con campos para el objeto "hotel.reservation". Los campos incluyen "guest_id" (un campo de selección para elegir un huésped), "room_id" (un campo de selección para elegir una habitación), "check_in_date" (fecha de entrada), "check_out_date" (fecha de salida), "precio_por_noche" (precio por noche de la habitación seleccionada), "precio_total" (precio total de la reserva), y dos botones: "Crear Reservación" y "Desocupar Habitación". El primer botón se utiliza para crear una nueva reserva y el segundo botón se utiliza para desocupar una habitación reservada.

```

ortafolio-main > views > reservaciones_view.xml
1  <?xml version="1.0" encoding="utf-8"?>
2  <odoo>
3  <record id="view_hotel_reservation_form" model="ir.ui.view">
4      <field name="name">hotel.reservation.form</field>
5      <field name="model">hotel.reservation</field>
6      <field name="arch" type="xml">
7          <form string="Reservación">
8              <sheet>
9                  <header>
10                     <field name="status" widget="statusbar"/>
11                 </header>
12                 <group>
13                     <field name="guest_id"/>
14                     <field name="room_id"/>
15                     <field name="precio_por_noche" readonly="1"/>
16                     <field name="check_in_date"/>
17                     <field name="check_out_date"/>
18                     <field name="precio_total" />
19                     <button name="onchange_room_id" string="Crear Reservación" type="object" class="btn-primary"
20                         attrs="{ 'invisible': [('status', '=', 'occupied')] }"/>
21                     <button name="onchange_desocupar" string="Desocupar Habitación" type="object" class="btn-primary"
22                         attrs="{ 'invisible': [('status', '!=', 'occupied')] }"/>
23                 </group>
24             </sheet>
25         </form>
26     </field>
27 </record>

```

La segunda parte del código define una vista de árbol con id "view_hotel_reservation_tree". Esta vista muestra una lista de todas las reservas existentes en una tabla con columnas para los campos "guest_id", "room_id", "check_in_date", "check_out_date", "precio_por_noche" y "precio_total".

```

26
27 <record id="view_hotel_reservation_tree" model="ir.ui.view">
28     <field name="name">hotel.reservation.tree</field>
29     <field name="model">hotel.reservation</field>
30 <field name="arch" type="xml">
31     <tree string="Reservaciones">
32         <field name="guest_id"/>
33         <field name="room_id"/>
34         <field name="check_in_date"/>
35         <field name="check_out_date"/>
36         <field name="precio_por_noche"/>
37         <field name="precio_total"/>
38     </tree>
39 </field>
40 </record>
41

```

La última parte del código define una acción con id "action_hotel_reservations" que se utiliza para mostrar las reservas en Odoo. Cuando el usuario selecciona esta acción, se abre una ventana que muestra la lista de reservas utilizando la vista de árbol y formulario definida anteriormente.

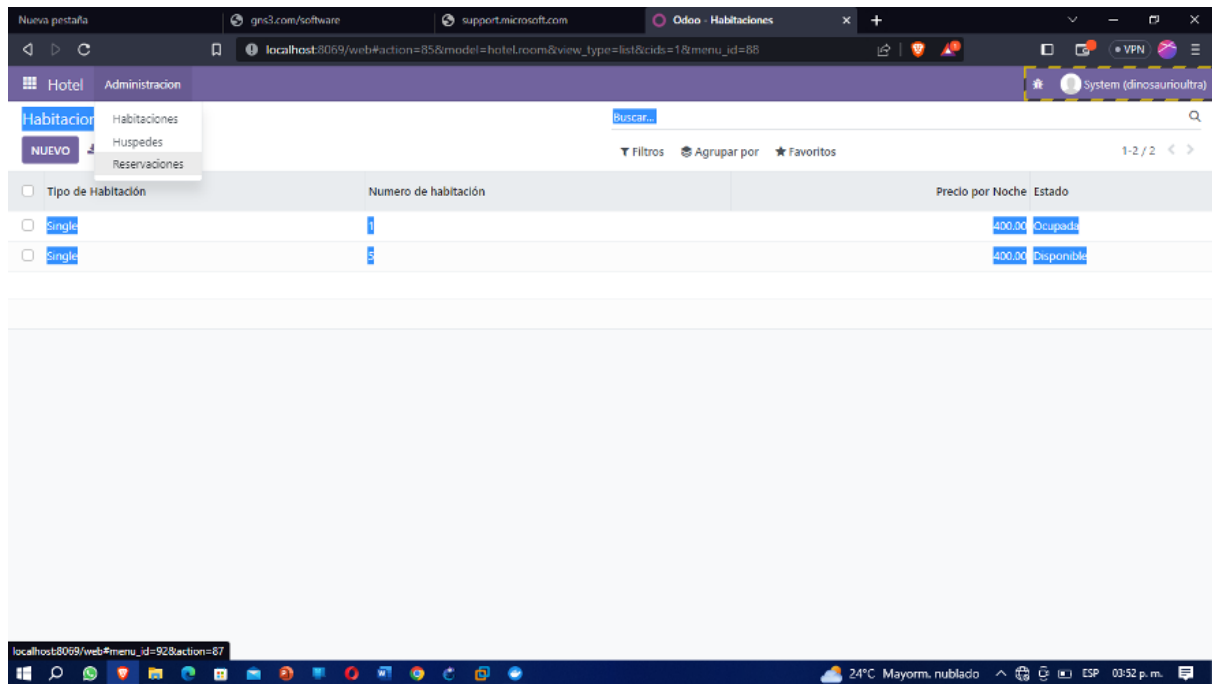
```

41
42 <record id="action_hotel_reservations" model="ir.actions.act_window">
43     <field name="name">Reservaciones</field>
44     <field name="res_model">hotel.reservation</field>
45     <field name="view_mode">tree,form</field>
46 </record>
47 </odoo>

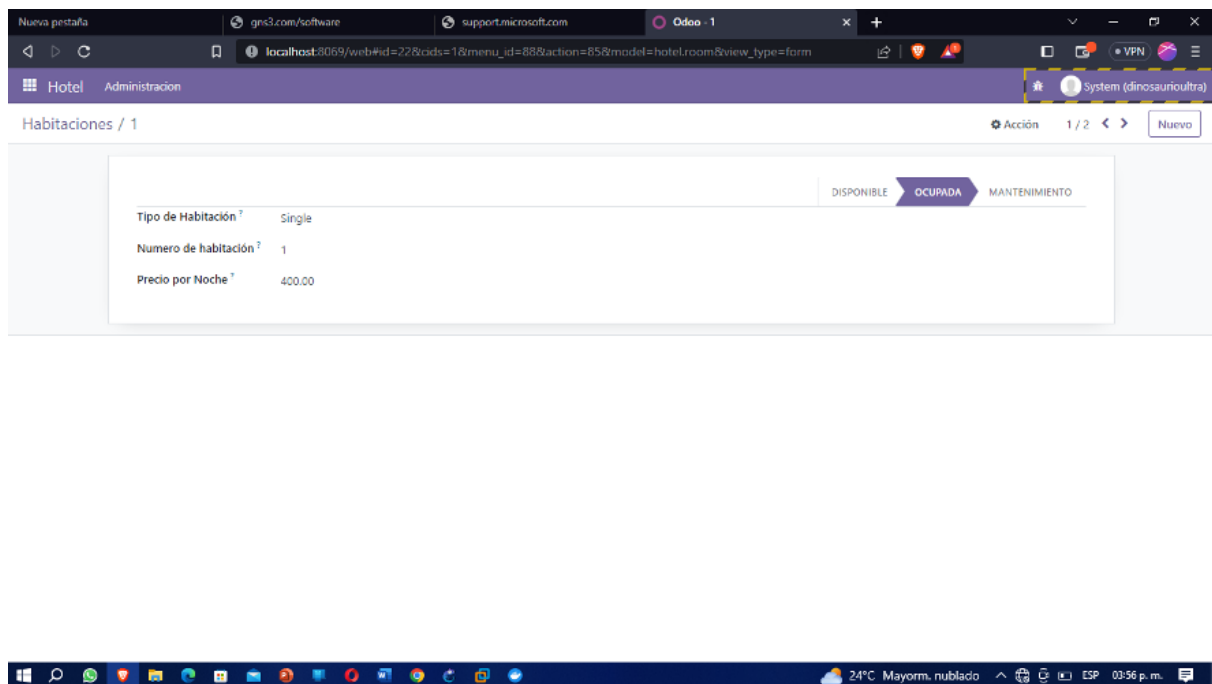
```


6 Funcionamiento

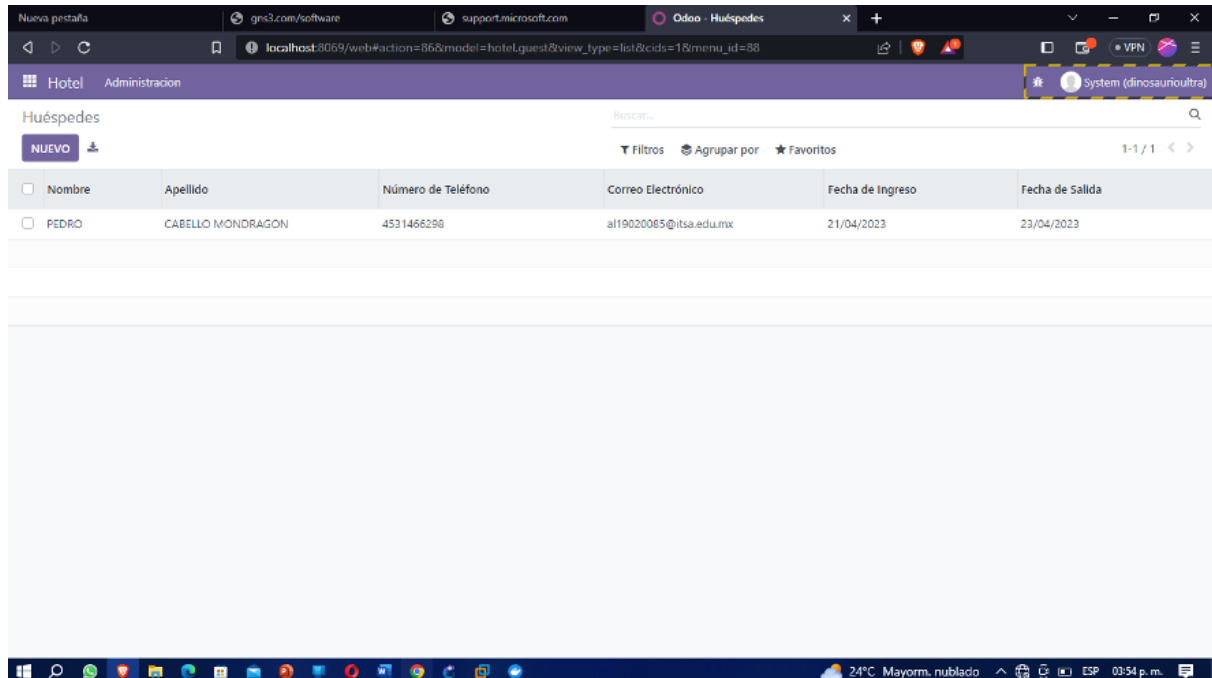
e



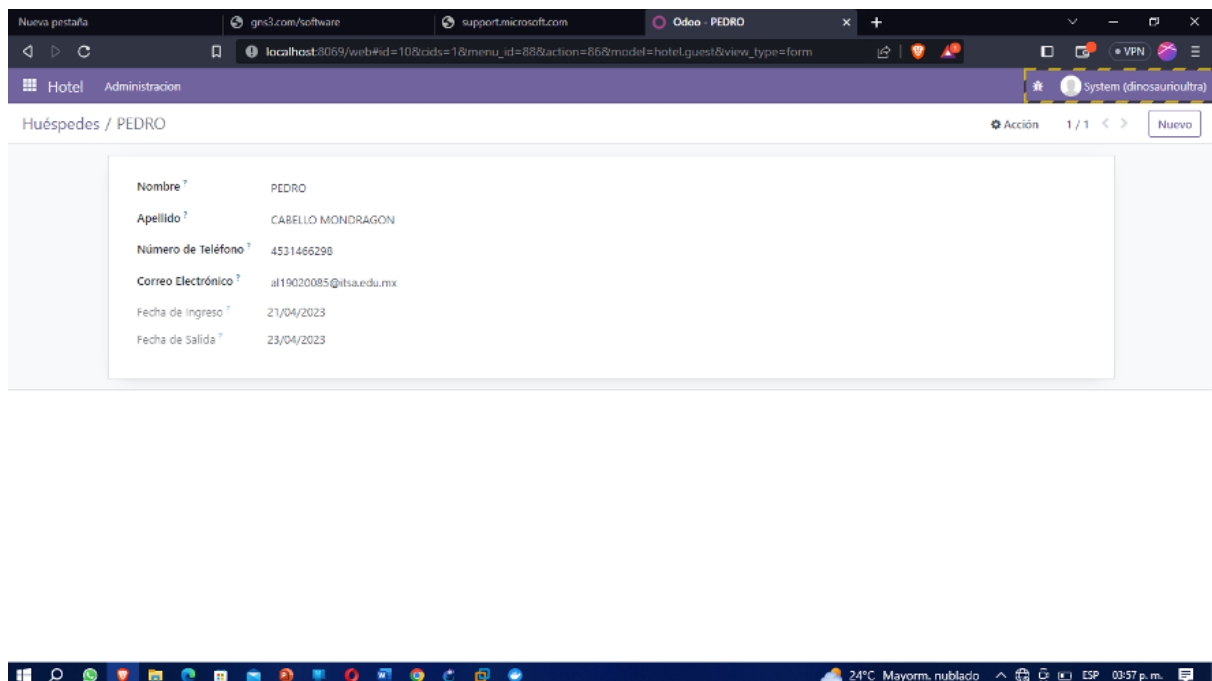
Esta ventana sirve para ver las habitaciones creadas



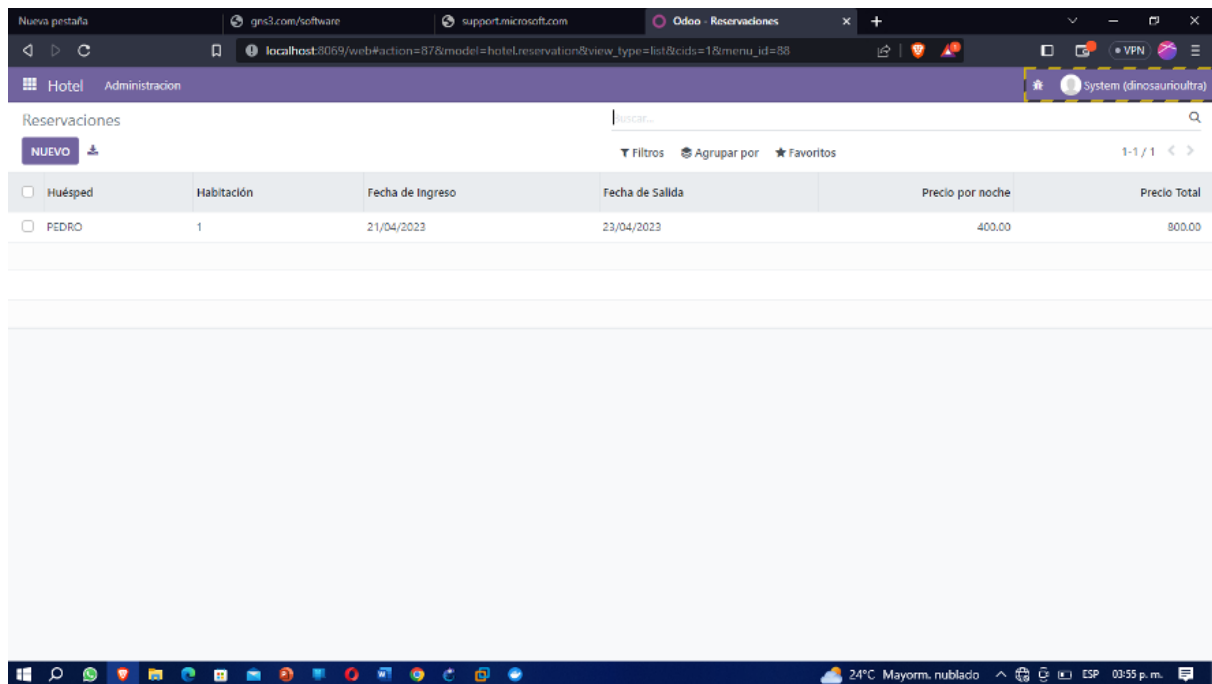
Esta ventana es donde podemos agregar una nueva habitación



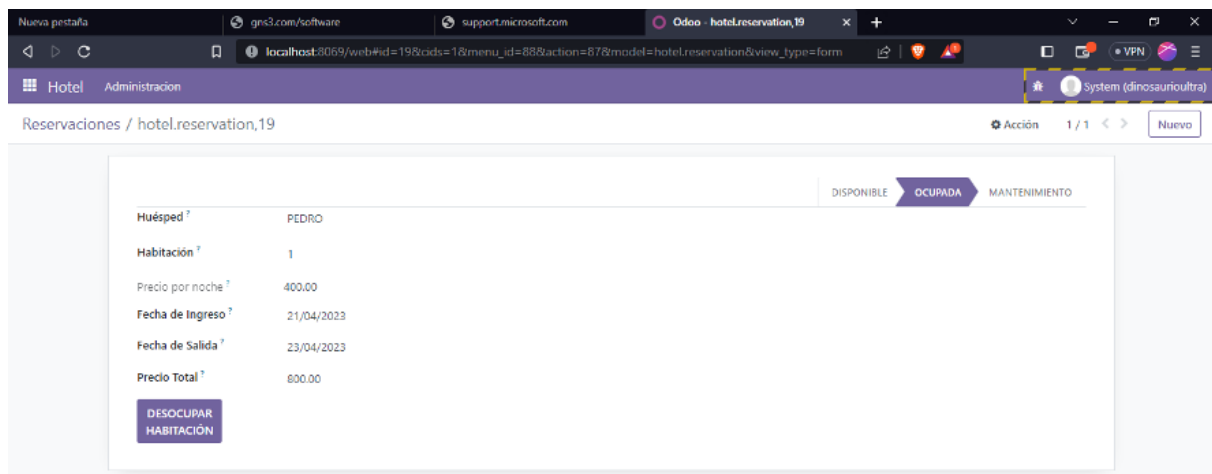
Esta ventana es donde podemos ver los huéspedes creados.



En esta ventana creados y administrados nuestros huéspedes



Aquí es donde creamos nuestra reservación, ten en cuenta que para crear una reservación se necesita tener antes un huésped y una habitación creada.



Aquí es donde administramos nuestras reservaciones y en donde podemos decidir si desocupar una habitación o no.

7 Conclusión:

Como visión general del proyecto de gestión del hotel en Odoo, donde se describen los diferentes aspectos clave del desarrollo. Se explica la importación de módulos y se detallan los datos definidos en el manifiesto del módulo. También se

presentan los modelos de datos para las habitaciones, huéspedes y reservaciones, junto con sus respectivos campos y métodos asociados. Además, se proporciona información sobre las vistas y acciones definidas en el proyecto, que se utilizan para mostrar y manipular los datos en la interfaz de usuario. Se muestra una página del hotel en funcionamiento, donde se presentan los apartados de habitaciones, huéspedes y reservaciones. En cuanto al método `_computr_status`, se explica cómo interactúa a través de las instancias de reserva y actualiza el campo `status` de cada reserva con el valor del campo `estado` del objeto `room_id` relacionado. A grandes rasgos, obtenemos una descripción completa de varios aspectos del proyecto de gestión de hoteles en Odoo, lo que permite tener una comprensión general de cómo se estructuran los modelos de datos, las vistas y las acciones, y cómo se implementa la lógica de actualización de campos en el caso específico del método `_computr_status`.

haber podido trabajar en equipo con este tipo de práctica nos ayudó bastante a poder desarrollar habilidades y poder mejorar aún más el manejo de una nueva herramienta o tecnología la cual es odoo la cual es muy usada para el manejo y creación de software de tipo empresarial en este caso el desarrollo de la práctica hotel nos brindó la oportunidad de poder entender cómo funciona un hotel y ver la forma de poder crear un software hecho a su medida para su uso exacto cuidando cada detalle como si de hacer un pastel de boda se tratara creemos que estamos muy satisfechos con lo que hemos echo