Universidad Castilla-La Mancha

# Odoofly – Memoria de prácticas

Sistemas de Información Empresariales

Curso 2019-2020

Alberto Novillo Chinchilla

miércoles, 8 de julio de 2020

# Índice

# Índice de Tablas



# Índice de Figuras

# Introducción

LIonsBussinesAir es una empresa internacional que organiza vuelos para el mundo de los negocios. La empresa cuenta con una flota de aviones (que cada vez crece más) operando en todo el mundo, realizando trayectos donde las empresas que son socias pueden mandar a sus empleados a cualquier lugar del mundo con un precio más reducido. La misma empresa mantiene relaciones con una lista de aeropuertos repartidos por todo el mundo donde los aviones pueden operar, como realizar embarques.

Debido al exitoso trayecto de esta joven empresa, se ha solicitado centralizar todas las operaciones y bases de datos en un ERP, concretamente en un módulo de Odoo v8. En el módulo, se gestionará una serie de requisitos (que serán especificados más adelante en este mismo documento) definidos a través de una serie de modulos, una serie de vistas, así como un pequeño programa en Python que conecte con la base de datos con una muestra de funciones a ampliar en un futuro.

## Requisitos del cliente

La empresa cliente ha solicitado que el módulo cumpla con, como mínimo, los siguientes requisitos:

1. Gestión de los aeropuertos con los que mantiene un contrato.
2. Gestión de los aviones que operan bajo la empresa.
3. Creación, gestión y mantenimientos de los trayectos que se realizan en la empresa, incorporando el avión y los aeropuertos

## Especificación detallada de los modelos utilizados

Para el desarrollo del módulo se han utilizado los siguientes modelos

**MODELO 1: AEROPUERTO**

| | |
|---|---|
| **NOMBRE MODELO** | *odoofly.aeropuerto* |
| **DESCRIPCIÓN** | Este modelo contendrá la representación de los datos relativa a los aeropuertos que mantiene la empresa cliente. Almacenará el nombre del aeropuerto y el país en el que está ubicado.<br><br>No es necesario establecer reglas o lógica de negocio adicional en este modelo. |
| **CAMPOS** | - **name**<br>*fields.Char()*<br>Campo de tipo cadena de caracteres que da un nombre reconocible al aeropuerto (p.e. Aeropuerto de Madrid-Barajas Adolfo Suárez). Este campo es de tipo requerido.<br><br>- **pais**<br>*fields.Many2one()*<br>Campo de tipo relación muchos a uno (un aeropuerto pertenece a un país; un país puede tener varios aeropuertos). Relacionado con el modelo *res.country* que corresponde a los países que almacena el sistema (p.e. España). |

*Tabla 1: Modelo1: Aeropuertos*

**MODELO 2: AVIÓN**

| | |
|---|---|
| **NOMBRE MODELO** | *odoofly.avion* |
| **DESCRIPCIÓN** | Este modelo contendrá la representación de los datos relativa a cada avión que la empresa cliente posee. Cada avión debe de poseer un nombre, un modelo y un numero de asientos. Además, cada avión está relacionado con un/unos trayecto/s.<br><br>No es necesario establecer reglas o lógica de negocio adicional en este modelo. |

| | |
|---|---|
| **CAMPOS** | - **name**<br>*fields.Char()*<br>Campo de tipo cadena de caracteres que da un nombre para poder identificar al avión (p.e. AACU).<br>Este campo es de tipo requerido.<br><br>- **numeroAsientos**<br>*fields.Integer()*<br>Campo de tipo entero que almacena el valor de cada asiento que puede ser ocupado por un pasajero durante un vuelo.<br>Este campo es de tipo requerido.<br><br>- **modelo**<br>*fields.Char()*<br>Campo de tipo cadena de caracteres que indica el tipo de modelo que es el avión (p.e Airbus 220).<br>Este campo es de tipo requerido.<br><br>- **trayectos_ids**<br>*fields.One2many()*<br>Campo de tipo relación uno a muchos (un trayecto usa un avión; un avión puede ser usado por varios trayectos). Relacionado con el modelo *odoofly.avion* que corresponde a los trayectos que almacena el sistema. |

*Tabla 2: Modelo 2: Avión*

**MODELO 3: TRAYECTO**

| | |
|---|---|
| **NOMBRE MODELO** | *odoofly.trayecto* |
| **DESCRIPCIÓN** | Este modelo contendrá la representación de los datos relativa a cada trayecto que la empresa cliente realice. Cada trayecto debe de poseer un nombre, una fecha y hora, tanto de salida como de llegada, así como almacenar el aeropuerto del que salió y al que arribó. Además, cada trayecto dispone de un avión (odoofly.avion) y de pasajeros.<br><br>Este modelo contiene alguna lógica de negocio adicional. |
| **CAMPOS** | - **name**<br>*fields.Char()*<br>Campo de tipo cadena de caracteres que da un nombre para poder identificar al trayecto.<br>Este campo es de tipo requerido.<br><br>- **fecha_salida**<br>*fields.Date()*<br>Campo de tipo date que almacena la fecha (día, mes y año) a la que el vuelo despegó.<br>Este campo es de tipo requerido. |

- **fecha_llegada**
  *fields.Date()*
  Campo de tipo date que almacena la fecha (día, mes y año) a la que el vuelo aterrizó.
  Este campo es de tipo requerido.

- **hora_salida**
  *fields.Char()*
  Campo de tipo cadena de caracteres que indica la hora y minutos a la que el vuelo despegó.
  Este campo es de tipo requerido.

- **hora_llegada**
  *fields.Char()*
  Campo de tipo cadena de caracteres que indica la hora y minutos a la que el vuelo aterrizó.
  Este campo es de tipo requerido.

- **origen**
  *fields.Many2one()*
  Campo de tipo relación muchos a uno (un trayecto tiene un aeropuerto de origen; un aeropuerto de origen puede haberlo sido por varios trayectos). Relacionado con el modelo *odoofly.aeropuerto* que corresponde a los aeropuertos que almacena el sistema (p.e. Aeropuerto de Madrid-Barajas Adolfo Suárez)..

- **destino**
  *fields.Many2one()*
  Campo de tipo relación muchos a uno (un trayecto tiene un aeropuerto de destino; un aeropuerto de destino puede haberlo sido por varios trayectos). Relacionado con el modelo *odoofly.aeropuerto* que corresponde a los aeropuertos que almacena el sistema (p.e. Aeropuerto Charles de Gaulle).

- **pasajeros**
  *fields.Many2many()*
  Campo de tipo relación muchos a muchos (un trayecto tiene varios pasajeros; un pasajero puede volar en varios trayectos). Relacionado con el modelo *odoofly.aeropuerto* que corresponde a los aeropuertos que almacena el sistema.
  Los pasajeros no podrán ser de tipo empresas.

- **avion_id**
  *fields.Many2one()*
  Campo de tipo relación muchos a uno (un trayecto utiliza un avión; un avión puede ser utilizado sido por varios trayectos). Relacionado con el modelo *odoofly.avion* que corresponde a los aviones que almacena el sistema.

| | |
|---|---|
| | - **asientos_disponibles**<br>*fields.Integer()*<br>Campo de tipo entero que almacena un entero indicando el número de asientos disponibles en un trayecto.<br>Este campo es de tipo calculado, indicando el siguiente valor:<br><br>*asientos_disponibles = avion.numeroAsientos – len(pasajeros)*<br><br>siendo len() la suma total del número de pasajeros. |
| **REGLAS NEGOCIO** | - **_verify_valid_date**<br>*Restricción (constrain)*<br>Comprueba si la fecha de llegada es anterior a la fecha de salida. En caso afirmativo, se producirá un error de validación, imposibilitando guardar el modelo.<br><br>- **_verify_valid_airports**<br>*Restricción (constrain)*<br>Comprueba si el aeropuerto de origen y de destino son iguales. En caso afirmativo, se producirá un error de validación, imposibilitando guardar el modelo.<br><br>- **_verify_valid_asientos**<br>*Restricción (constrain)*<br>Comprueba si los asientos disponibles de un trayecto son negativos. En caso afirmativo, existen más pasajeros que asientos disponibles, por lo que se producirá un error de validación, imposibilitando guardar el modelo. |

*Tabla 3: Modelo 3 Trayecto*

Además de la creación de estos modelos, se ha modificado el modelo res.parner para incluir un nuevo campo many2many llamado vuelos, correspondiente con la lacion many2many con el modelo trayectos (figura 10).

## Diagrama de las relaciones entre modelos

En la figura 1, se pueden ver las relaciones existentes entre los modelos, así como la cardinalidad de estas relaciones. Nótese que algunos modelos están oscurecidos, lo cual significa que estos modelos son exteriores al módulo desarrollado.

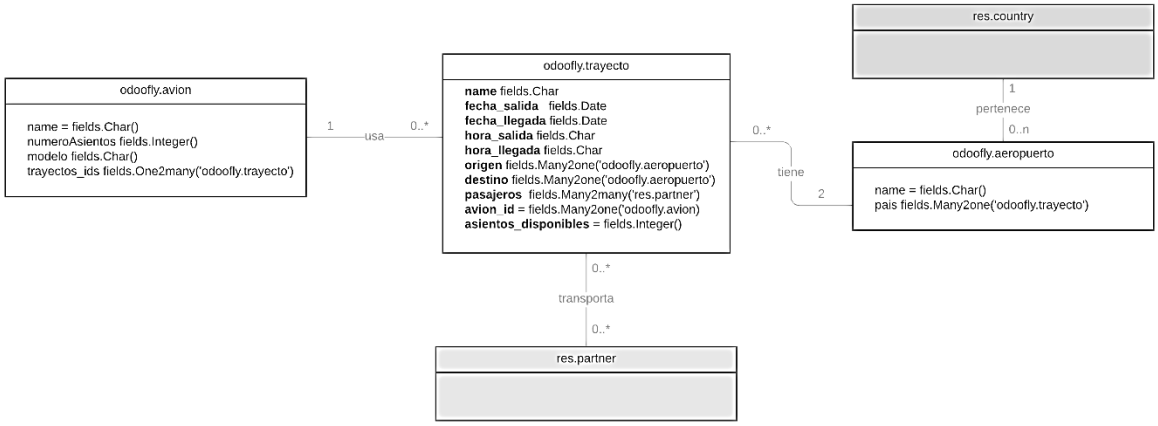


*Figura 1 Relación entre modelos*

# Vistas del módulo

Los datos serán representados en un modulo de Odoo accesible desde el menú principal. Desde este menú, se podrán acceder a los aeropuertos, aviones y trayectos a través de un menú lateral. Además, cada vez que se acceda al modelo, lo primero que se mostrara es el listado de aviones, incluyendo su nombre, el modelo y el numero de asientos de cada avión como se puede observar en la figura 2.

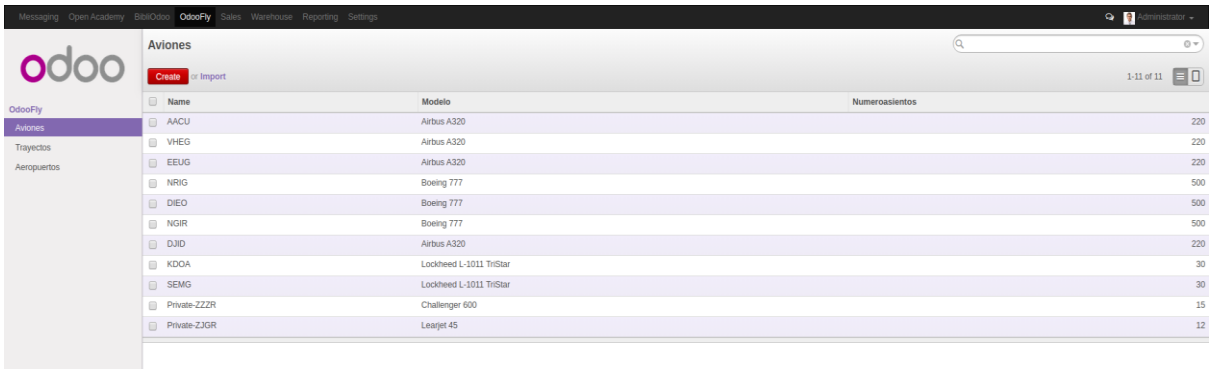


*Figura 2: Listado de Aviones*

Si accedemos a un avión, se nos mostrará un formulario donde podremos ver los datos de dicho avión (nombre, modelo y número de asientos), así como una tabla donde se muestran los trayectos que han sido realizados o están programados que ha efectuado el avión, incluyendo el nombre, el origen, destino y fecha de salida del vuelo (Figura 3).

En la segunda opcion del menú, podremos encontrar los trayectos de los que guarda informacion el sistema, incluidos los ya realizados y los que estan programados en un futuro.

En esta vista del menu, podremos encontrar una lista de trayectos, clasificandolos por el nombre del vuelo, el nombre del avión en el que se navega y la fecha y hora de salida (Figura 4).

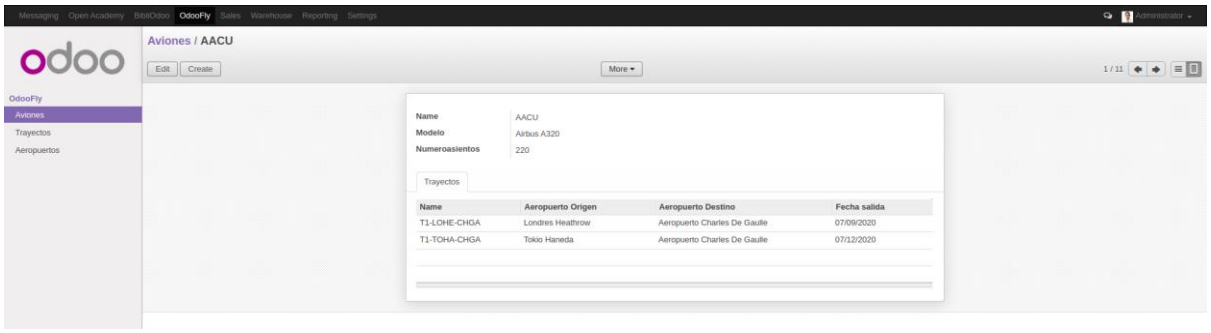


*Figura 3: Formulario del interior de un avión*

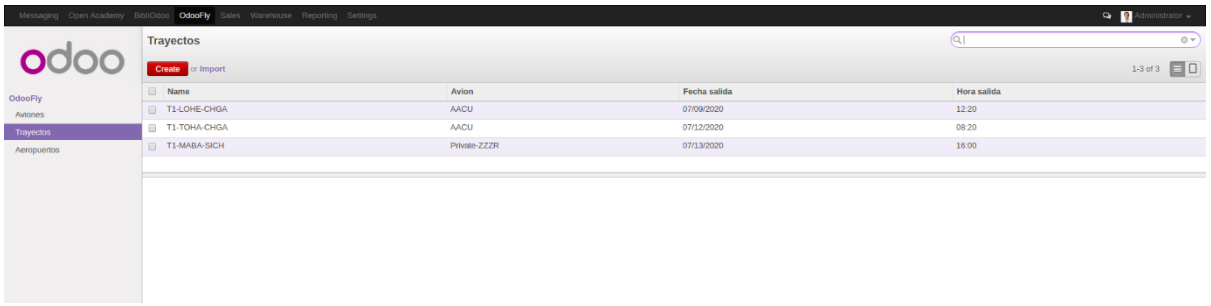


*Figura 4: Listado de trayectos*

Si pulsamos en un vuelo, podremos ver un formulario donde aparecerán los datos del vuelo. Estos datos están divididos en 3 partes: Información general (que incluye el nombre del vuelo, fecha y hora de llegada y de despegue, el avión con el que se realiza el vuelo y el numero de asientos disponibles.

Nótese que, si no se selecciona el avión, el numero de asientos disponibles será 0. También es importante señalar que el número de asientos puede ser negativo. En este caso, el trayecto nunca podrá guardarse. En cuanto a las otras dos partes de la información, estarán divididas en dos pestañas. La primera mostrará el origen y destino del vuelo (figura 5) y la segunda pestaña mostrará una lista de todos los pasajeros a bordo, incluyendo su nombre, teléfono y email si disponen de él (figura 6).

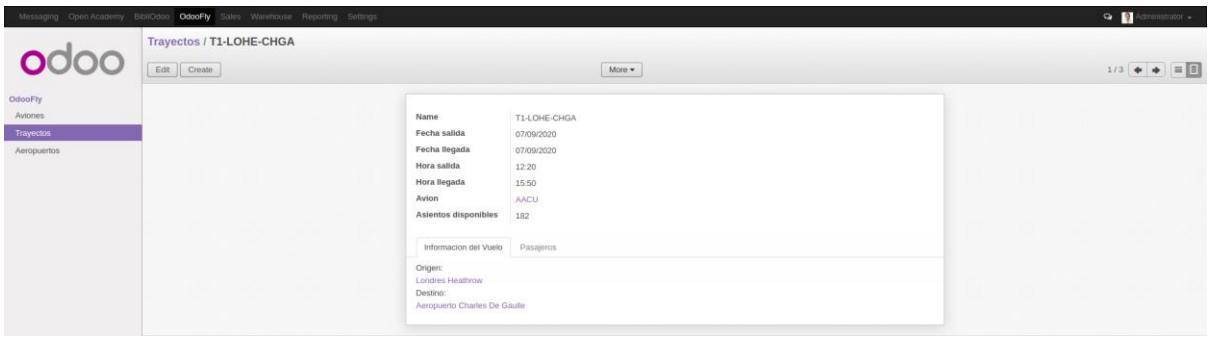


*Figura 5: Interior de un trayecto - Información del vuelo*

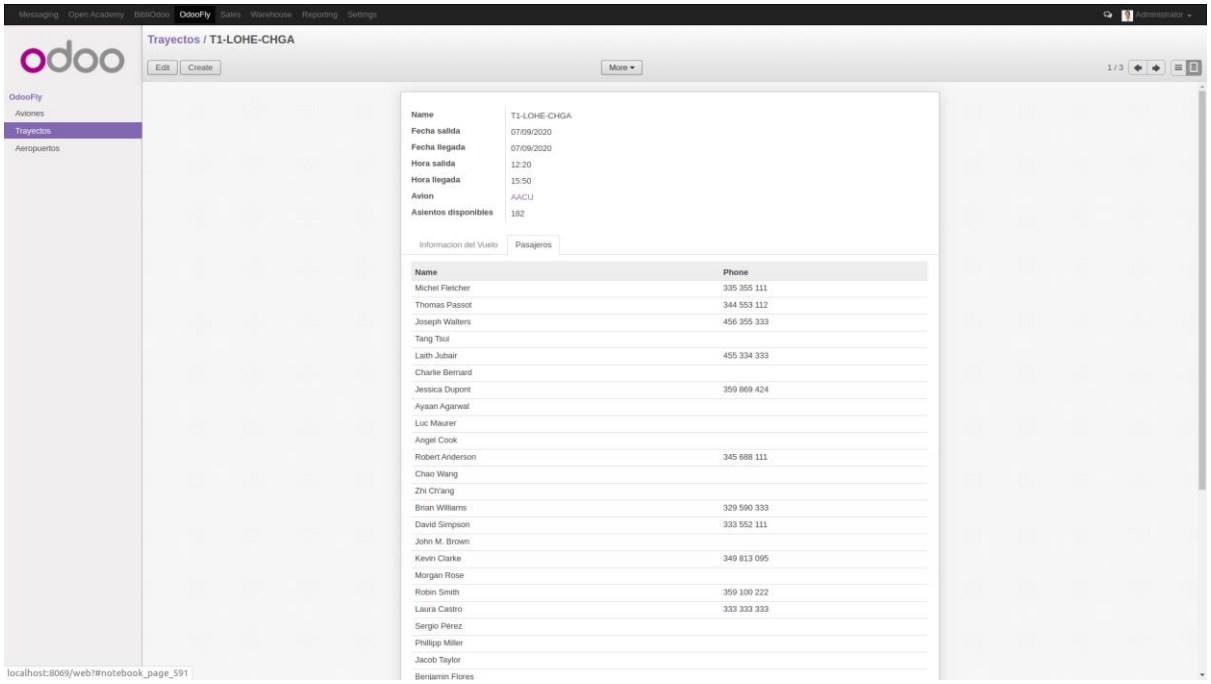


*Figura 6: Interior de un trayecto – Pasajeros*

Además, es importante destacar que en la creación de una nueva instancia en el modelo a través de esta vista, se harán una serie de comprobaciones antes de guardarlas. Estas comprobaciones serán:

- Comprobar si los aeropuertos de llegada y de salida son iguales.
- Comprobar si la fecha de llegada es posterior a la fecha de salida.
- Comprobar si el numero de pasajeros excede al total de asientos del avión.

En caso de que alguna comprobación no se cumpla, se avisará al usuario de forma grafica como puede verse en la figura 7.

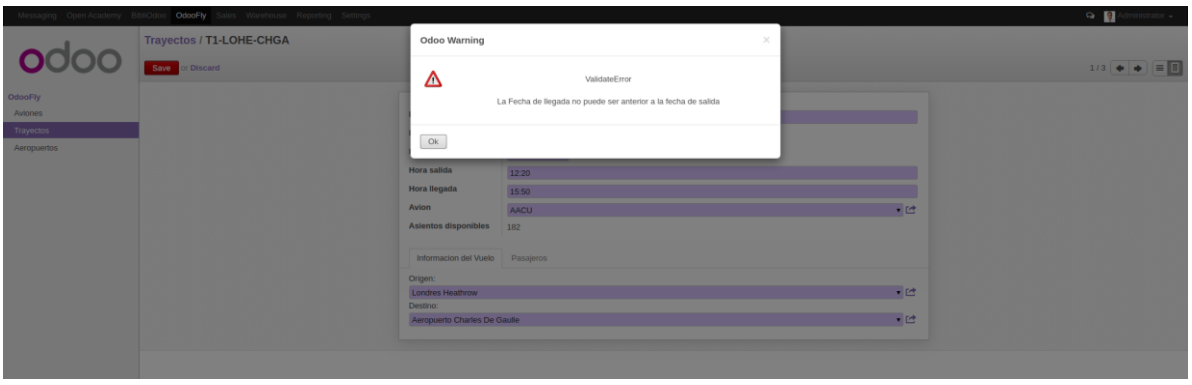


*Figura 7: Aviso error de fecha*

En la última opción del menú, encontraremos el apartado de aeropuertos. Como en los anteriores, al pulsar esta opción, accederos a una lista de aeropuertos, donde se nos mostrará el nombre de la instalación (figura 8). Si pulsamos sobre un aeropuerto, también aparecerá el país al lado del nombre (figura 9).

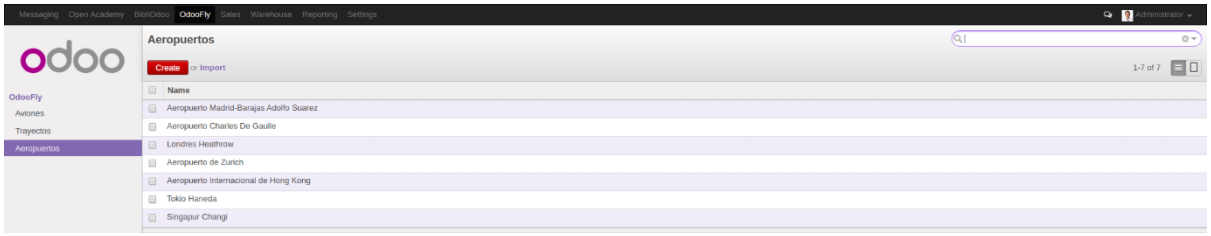


*Figura 8: Listado de aeropuertos*


*Figura 9: Interior de un aeropuerto*

Por último, podremos ver los vuelos que ha realizado un pasajero (*res.partner*) en la vista de su información. Se ha agregado una nueva pestaña en su vista llamada vuelos, donde se muestra el historial de vuelos, con el nombre del vuelo, el avión y la fecha y hora de salida.

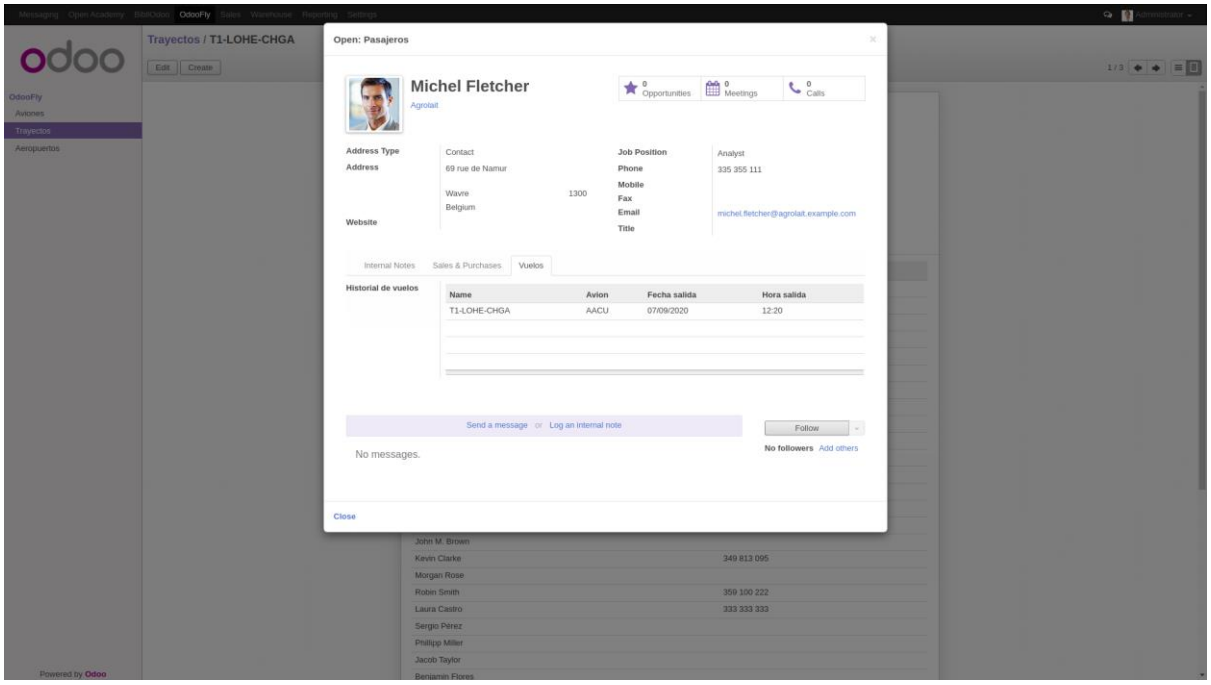


*Figura 10: Interior de un pasajero (res.partner)*

# Anexo I: Funcionamiento del código Python de consulta

Antes de ejecutar el código Python debemos de asegurarnos que lo siguiente:

1. La base de datos esta alojada y accesibles en el localhost y en el puerto 8069.
2. La base de datos se llama odoo.
3. El usuario y contraseña es admin para ambos.

Una vez que se haya conseguido la conexión, accederemos a un simple menú donde se nos indicarán las opciones disponibles y cual queremos elegir (figura 11).

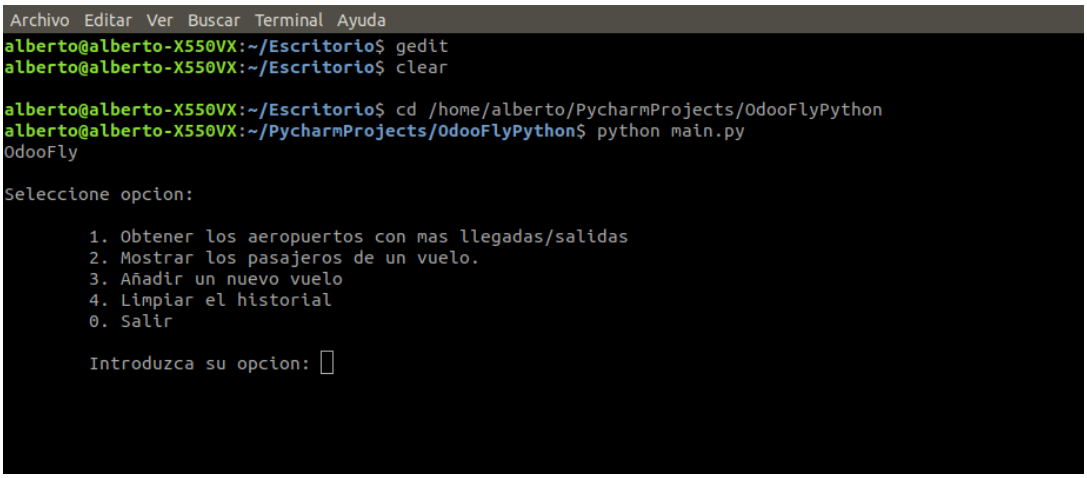


*Figura 11: Menú Python consultas*

Las opciones del menú son:

1. Obtener los aeropuertos con mas llegadas/salidas
2. Mostrar los pasajeros de un vuelo.
3. Añadir un nuevo vuelo
4. Limpiar el historial
0. Salir.

Introduciremos el valor 1 para ver que aeropuertos son los que más llegadas y más salidas tienen (Figura 12).

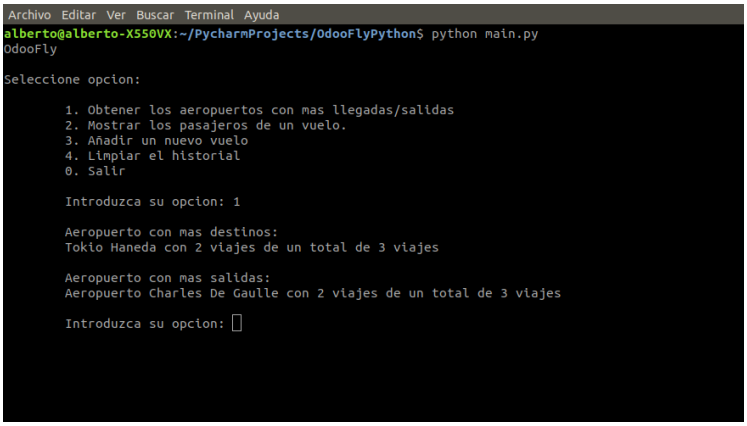


*Figura 12: Resultados aeropuertos con más llegadas/salidas*

```
def aeropuertosMasVisitado():
    aeropuertos_ids = object.execute_kw(dbname, uid, user_passwd, 'odoofly.trayecto', 'search', [[]])

    aeropuertos_destino = object.execute_kw(dbname, uid, user_passwd, 'odoofly.trayecto', 'read', [aeropuertos_ids], {'fields': ['destino']})
    aeropuertos_origen = object.execute_kw(dbname, uid, user_passwd, 'odoofly.trayecto', 'read', [aeropuertos_ids], {'fields': ['origen']})
    listaaux1 = []
    listaaux2 = []
    listadestinos = []
    listaorigenes = []

    for x in aeropuertos_destino:
        airport = x
        listaaux1.append(airport['destino'])
    for x in aeropuertos_origen:
        airport = x
        listaaux2.append(airport['origen'])
    for destinos in listaaux1:
        listadestinos.append(destinos[1])
    for origenes in listaaux2:
        listaorigenes.append(origenes[1])
    if len(listadestinos) != 0 or len(listaorigenes) != 0 :
        print "\n\tAeropuerto con mas destinos:"
        most_frequent(listadestinos)
        print "\n\tAeropuerto con mas salidas:"
        most_frequent(listaorigenes)
    else:
        print "\tNo se han establecido vuelos aun"
```

*Figura 13: Código de la consulta de aeropuertos más concurridos*

El funcionamiento del algoritmo es el que se puede observar en la figura 13. El algoritmo accede a la base de datos y en ella al modelo *airport*, obteniendo los nombres e ids de todos los aeropuertos de salida y llegada de todos los trayectos. Posteriormente se limpian los datos (dejando una lista con los nombres) y se busca el más repetido a través del método *most_frecuent*. Por ultimo se muestra por pantalla.

La siguiente opción del menú es la de mostrar los pasajeros de un vuelo (figura 14). Esta función requerirá que le pasemos el nombre del vuelo del que queremos obtener los pasajeros. Una vez obtenido, el programa accederá a la base de datos, buscando un trayecto con ese nombre.

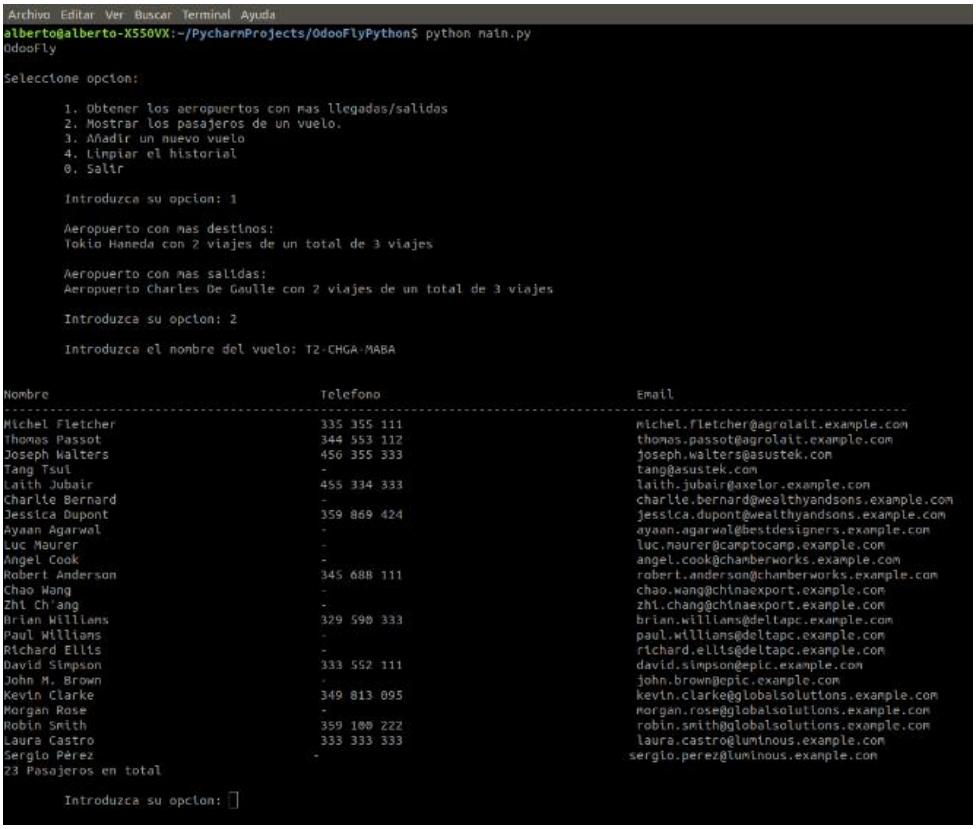


*Figura 14: Pasajeros de un vuelo*

El resultado obtenido será limpiado y convertido en una lista de Ids que corresponden a los ids de los pasajeros (res.partner). Para cada id, accederos a la base de datos y obtendremos su nombre, su teléfono y su email (figura 15). En el caso de que el email o el telefono del pasajero esté vacio, se mostrará con un -.

```
def mostrarPasajeros():
    #print "\n\tIntroduzca el nombre del vuelo: "
    nme = raw_input("\n\tIntroduzca el nombre del vuelo: ")
    print "\n"
    trayecto_id = object.execute_kw(dbname, uid, user_passwd, 'odoofly.trayecto', 'search_read', [[['name','=',nme]]], {'fields': ['pasajeros']})
    if len(trayecto_id) != 0:
        listaux= trayecto_id[0]
        lista_pasajeros = listaux['pasajeros']

        print '{:<40}  {:<40}  {:<40} '.format("Nombre", "Telefono", "Email" ) +"\n--------------------------------------------------------------------------------------------------------------------------------"

        for ps in lista_pasajeros:
            pasajero = object.execute_kw(dbname, uid, user_passwd, 'res.partner', 'search_read', [[['id','=',str(ps)]]], {'fields': ['name', 'phone', 'email']})
            aux = pasajero [0]
            if not aux['phone']:
                aux['phone'] = '-'
            if not aux['email']:
                aux['email'] = '-'

            print '{:<40}  {:<40}  {:<40} '.format(aux['name'].encode('utf-8') , aux['phone'].encode('utf-8') , aux['email'].encode('utf-8')  )

        print str(len(lista_pasajeros)) + " Pasajeros en total"

    else:
        print "Vuelo no encontrado"
```

*Figura 15: Código mostrarPasajeros*

La tercera opción del menú nos permitirá añadir un nuevo trayecto a la base de datos. Para ello se requieren los datos necesarios (nombre, hora y fecha tanto de llegada como de salida, id del avión y origen y destino del vuelo). Una vez tomados estos datos, se pedirá que se introduzca el nombre de

los pasajeros uno a uno hasta escribir FIN, lo que hará que se añada a la base de datos el nuevo vuelo. Es importante mencionar que las posibles excepciones que pudieran ocasionarse al añadir un nuevo vuelo (fecha de llegada posterior a la de salida, mas pasajeros que asientos, mismo aeropuerto de salida y origen) son controladas y capturadas antes de añadirlas a la base de datos, por lo que si se añade un dato incorrecto no se añadirán. Los resultados pueden verse en la figura 16.

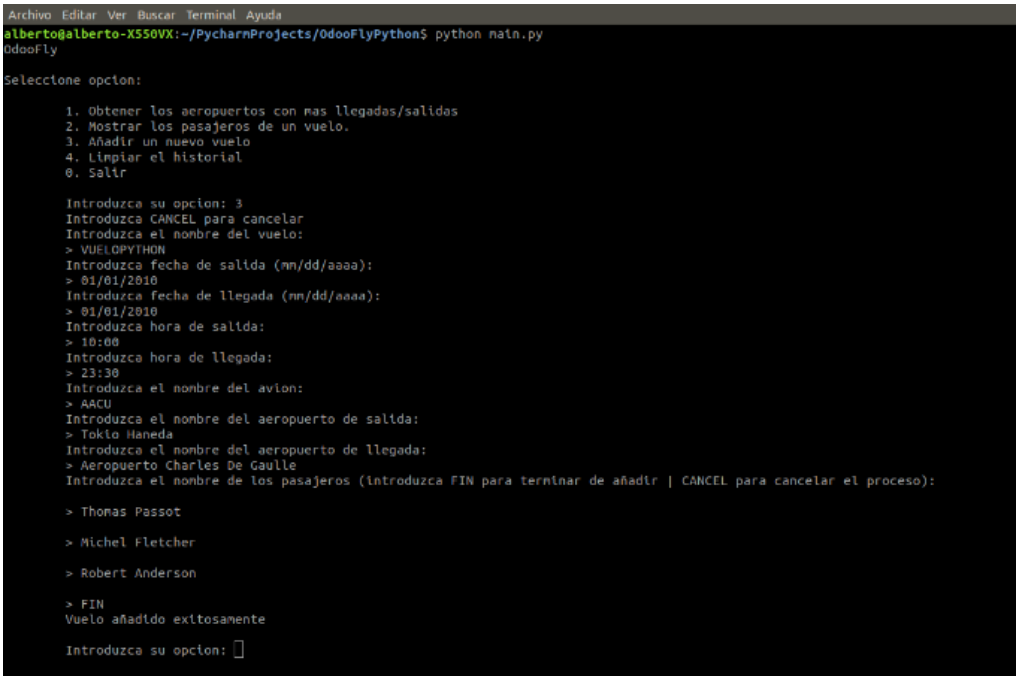


*Figura 16: Resultado de la función Añadir un Vuelo*

EL funcionamiento del algoritmo para añadir un nuevo vuelo es el siguiente:

1. Ponemos el programa a la escucha para recibir los datos del vuelo.
2. Comprobamos que los nombres del aeropuerto y del avión son correctos y existen en la base de datos. En caso afirmativo, obtenemos el id de estos datos y en otro caso finalizamos el programa.
3. Una vez obtenido los datos del vuelo, entramos en un bucle donde se recibe el nombre del pasajero a añadir. Para cada nombre, accedemos a la base de datos para comprobar si existe y en caso afirmativo, obtenemos su id.
4. Una vez obtenidos todos los datos, realizamos las comprobaciones de que son válidos (aeropuertos iguales, límite de pasajeros). Si lo son, procedemos a escribir en la base de datos una nueva instancia del modelo.

En la figura 17 se puede ver como una vez añadido el vuelo, podemos acceder a sus pasajeros a través de la opción 2:

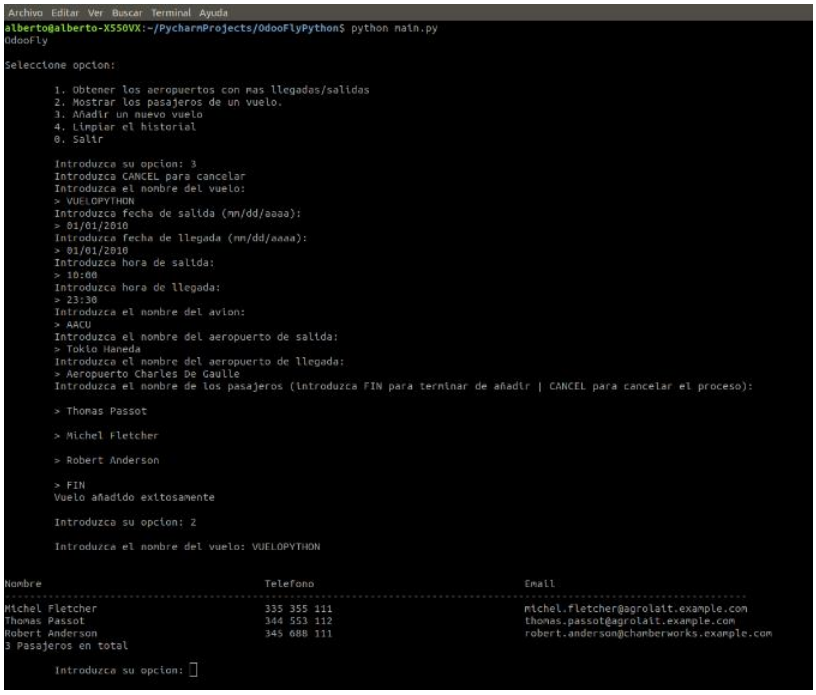


*Figura 17: Resultado de la función limpiar Historial*

La ultima opción (limpiar el historial) nos permite eliminar todos los vuelos cuya fecha de salida sea anterior a la fecha de hoy. Para ello, en el paso anterior indujimos una fecha muy anterior a la de hoy (2010). Pulsamos la opción 4 y nuestro vuelo añadido en el paso 3 nos aparece a punto de eliminar.

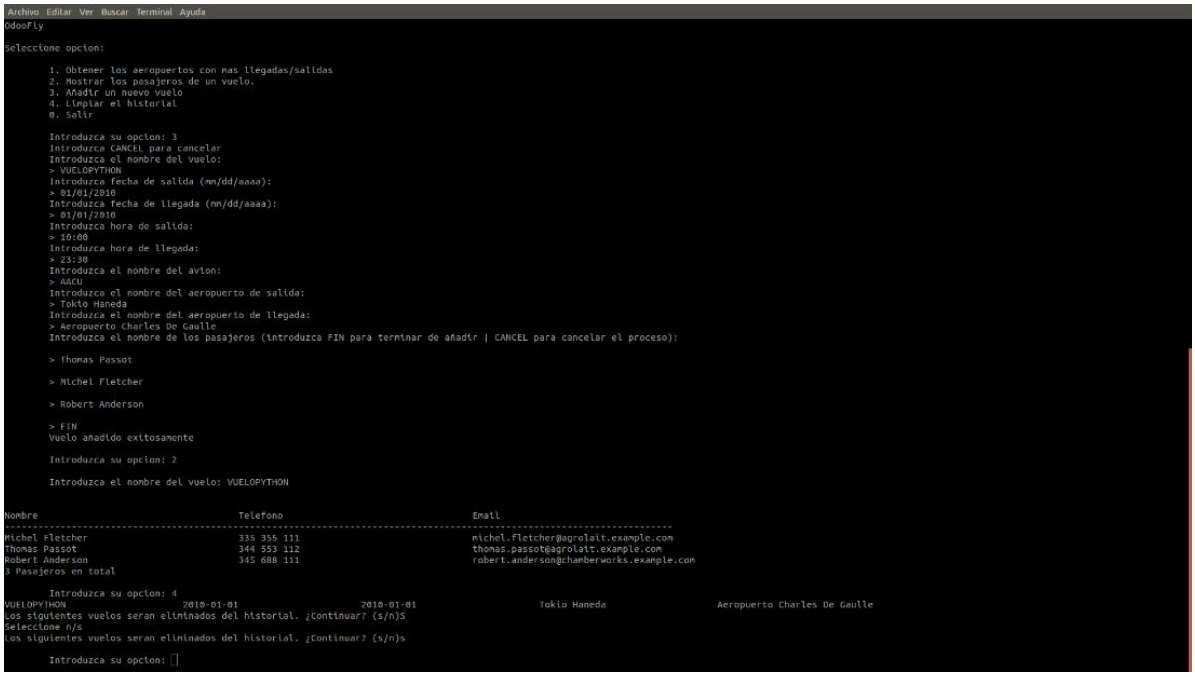


*Figura 18: Borrado de los vuelos anteriores a hoy*

Pulsamos s y el vuelo es eliminado satisfactoriamente (figura 18).

El funcionamiento de este paso es el siguiente:

1. Accedemos a la base de datos y obtenemos todas las ids de los vuelos cuya fecha sea anterior a la de hoy (librería time).
2. En el caso de que existan vuelos anteriores a hoy, tras la orden de eliminar por parte del usuario, se accederá a la base de datos para eliminar todos los vuelos cuyas ids correspondan a los obtenidos.

# Anexo II: Manual de Usuario – Funcionamiento del modulo

En este apartado se mostrara paso por paso como podemos crear un avion y asignarlo a un nuevo trayecto.

El primer paso, una vez instalado el modulo sera acceder al propio modulo a traves del menú (Figura 19, menú superior -> OdooFly).

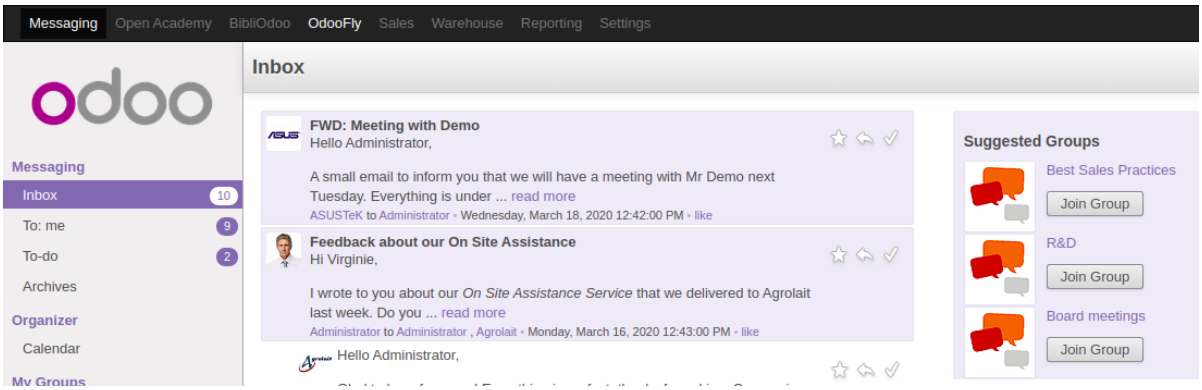


*Figura 19: Menú principal*

Una vez accedido, se mostrará el listado de aviones (Figura 20). Para crear un nuevo avión, pulsaremos el botón Create para crear un nuevo avión.

Se abrirá un formulario donde saldrán los campos de información del avión a rellenar (Figura 21). Tenga en cuenta de que todos los datos son obligatorios, por lo que no podrá guardar el avión si no se rellena adecuadamente todos los datos. Rellenaremos los campos con los siguientes datos:

1. Name → AVION-TUTORIAL
2. Modelo → MODELO-TUTORIAL
3. Numero de Asientos → 15

Por el momento, dejaremos el apartado de trayectos dentro del avión vacío, ya que en el siguiente punto nos centraremos en crear un nuevo trayecto. Pulsamos Guardar y comprobamos como nuestro nuevo avión ha sido añadido exitosamente.

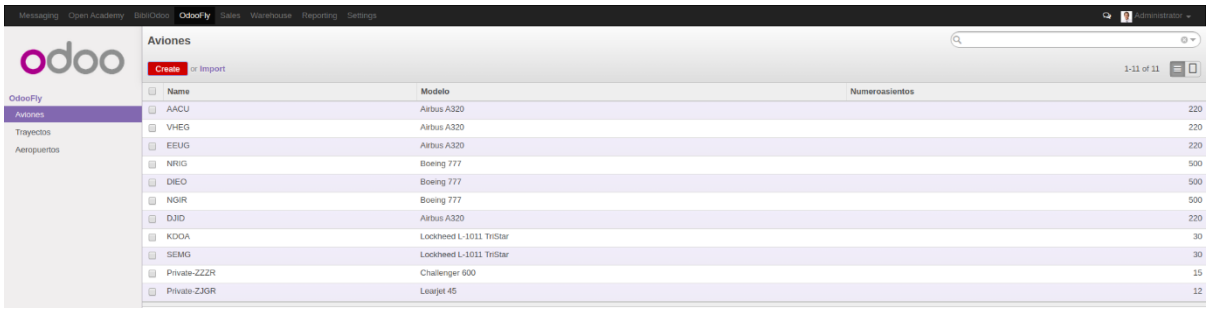


*Figura 20: Listado de Aviones*

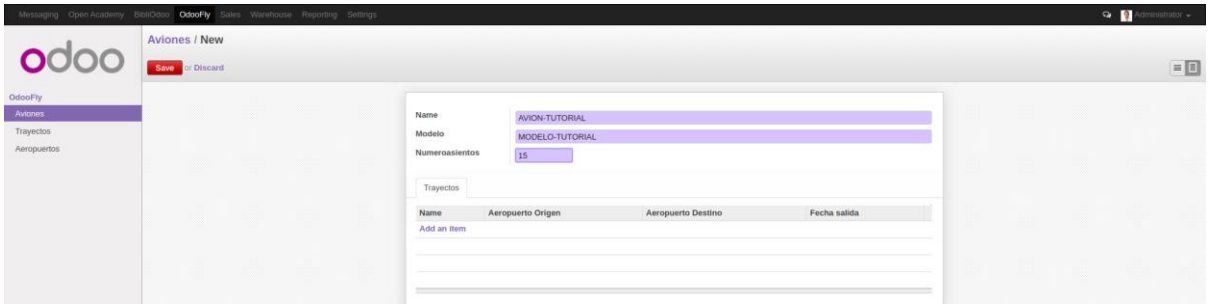


*Figura 21: Formulario datos del avión*

Una vez creado el avión, nos dirigiremos al apartado Trayectos a través del menú lateral de la izquierda. Aparecerá un listado de los trayectos registrados (Figura 22). Pulsamos el botón de Create para añadir un nuevo vuelo.

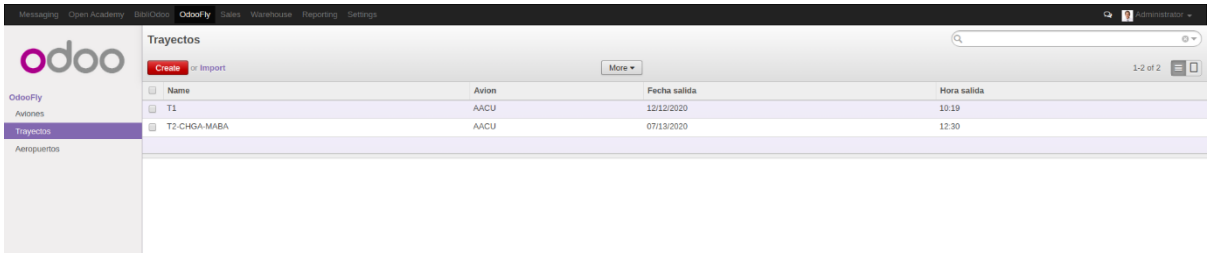


*Figura 22: Listado de Trayectos*

Una vez pulsado, aparecerá un formulario similar al anterior donde podremos rellenar los datos del vuelo (Figura 23).

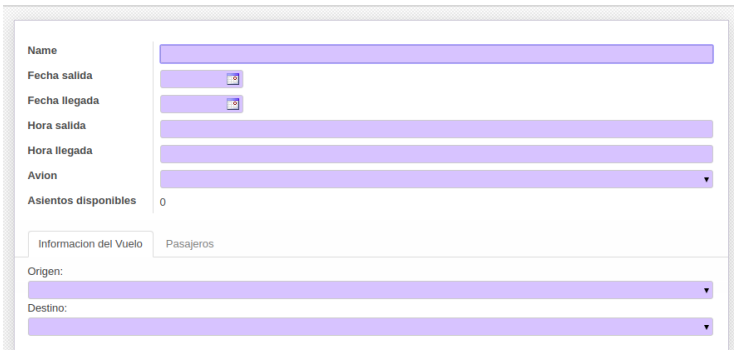


*Figura 23: Formulario de nuevo trayecto*

Empezaremos rellenando el nombre con el valor “Trayecto1-tutorial” y los demás datos con lo siguiente (figura 24):

1. Fecha de salida → 07/15/2020
2. Fecha de llegada → 07/15/2020
3. Hora de salida → 16:30
4. Hora de llegada → 19:50
5. Avión → AVION-TUTORIAL [1]

[1] AVION-TUTORIAL es el avión que hemos creado en el paso anterior. En caso de no aparecer, pulse la opción *search* y busque el avión.

6. Origen →Aeropuerto de Zúrich [2]
7. Destino → Londres Heathrow

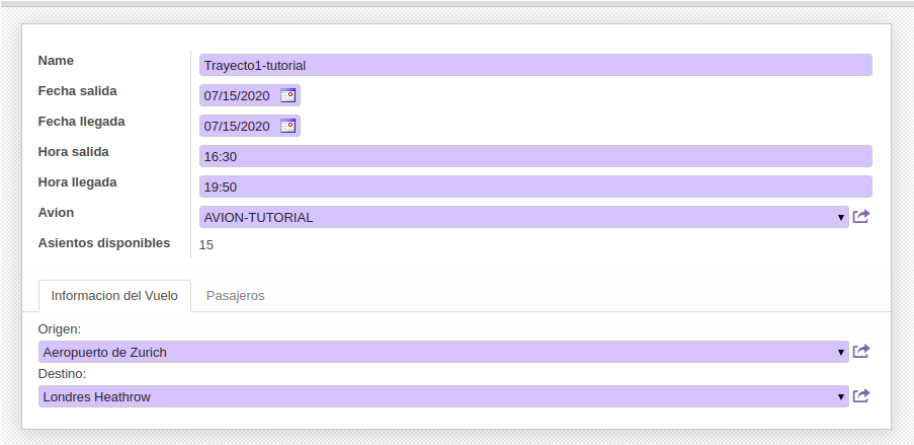


*Figura 24: Rellenado de datos en el formulario de trayectos*

Observe como el avion que hemos creado está en la lista de aviones disponibles y que los asientos que hemos introducido aparecen en el campo Asientos Disponibles. Pulsaremos ahora en la pestaña Pasajeros para empezar a añadir a los pasajeros que realicen el viaje. Pulsamos la opcion *add item* para empezar añadir personas (Figura 25).

Seleccionamos los pasajeros que realizaran el trayecto. Observe que no existe limite para seleccionar a los pasajeros. Seleccionaremos 18 pasajeros, por lo que el numero de asientos disponibles será -3 (Figura 26).

Para poder guardar satisfactoriamente, el número de asientos disponibles deberá de ser mayor o igual que 0, por lo que eliminaremos los 3 últimos pasajeros. Una vez hecho esto, podremos guardar el vuelo y comprobar que se ha añadido exitosamente a la lista de trayectos (figura 27).

[2] *Aeropuerto de Zúrich* y *Londres Heatrow* son aeropuertos que vienen cargados con los datos de demostración del módulo. Si no han sido cargados, deberemos de crear los dos aeropuertos que queramos que sean el origen y el destino. Para ello, debemos de acceder a la pestaña aeropuertos del menú lateral, botón créate, añadir el nombre y seleccionar el país al que pertenece y pulsar guardar. Recargar la pagina del formulario y comprobar que ya están disponibles.

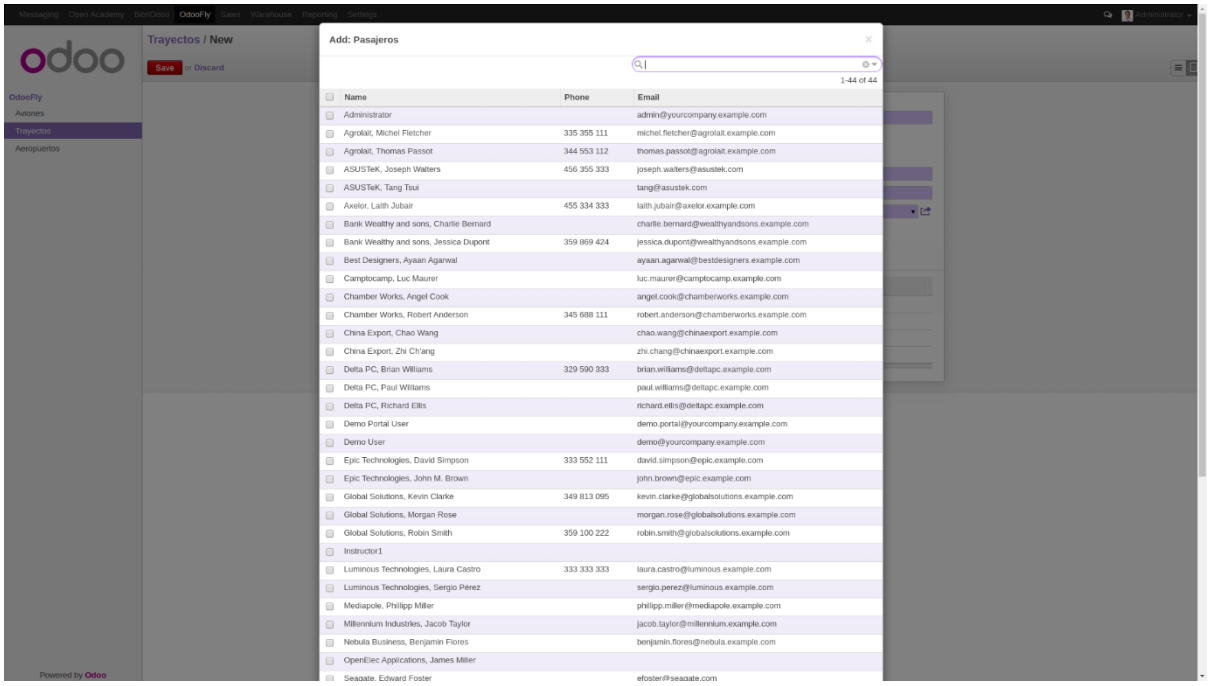


*Figura 25: Selección de pasajeros del trayecto*

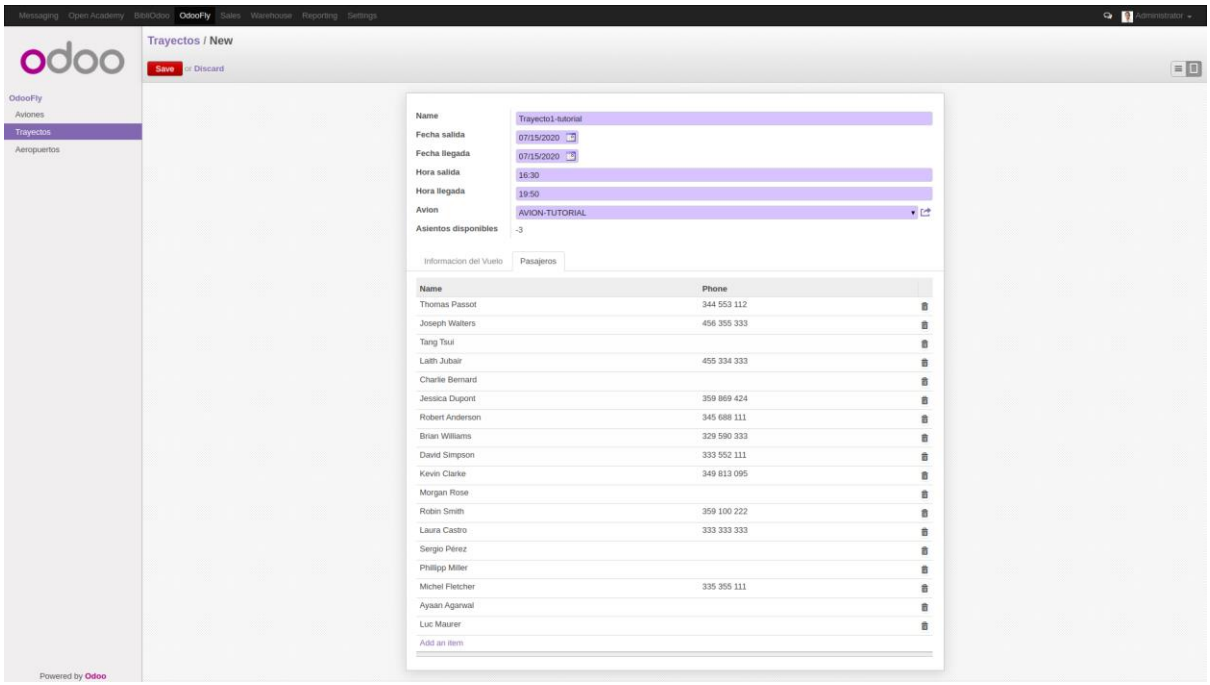


*Figura 26: Exceso de pasajeros (número de asientos -3)*

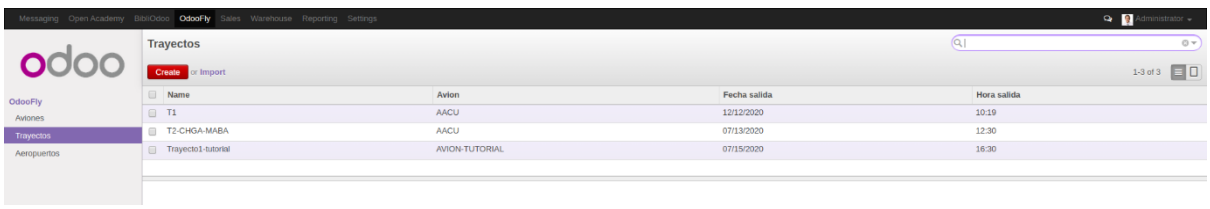


*Figura 27: Listado de trayectos con la última incorporación*