Practica4

May 24, 2020

1 Introducción a los Servicios Web

- ¿ Que es un servicio web ?

  Un servicio web es una tecnología que utiliza una serie de protocolos y estándares con la finalidad de que sea posible realizar un intercambio de datos entre aplicaciones, independientemente de los lenguajes de programación con los que estas han sido desarrolladas e independientemente de las plataformas sobre las que se están ejecutando.

- Características de los servicios web:

  1. Deben de poder ser localizables y accesibles a través de internet.
  2. Deben de contener una descripción de si mismos de forma que una aplicación externa pueda saber cual es la función de dicho servicio web y cuales son sus interfaces, pues estos últimos nos permitirán comunicarnos con ella.

- Situaciones en las que se emplean servicios web:

  1. Cuando se desean conectar aplicaciones muy diferentes.
  2. Para realizar una serie de actividades que permitan completar una tarea (workflows).

- ¿ Porqué es ventajoso construir programas cuyas arquitecturas de software están orientadas a servicios (SOA) ?

  1. Se reduce la cantidad de código en la capa de negocio, pues su lógica puede mediante estos.
  2. Su empleo supone un bajo acoplamiento entre módulos de código.
  3. Son externos e independientes con respecto a nuestra plataforma, por lo que no tendremos que aplicarles mantenimiento ni cambio alguno, a menos que estos realizen cambios en sus interfaces.
  4. Permiten construir sistemas altamente escalables, es decir, permiten que el tamaño de los sistemas software aumente manteniendo la calidad en lo referente a la arquitectura y la funcionalidad implementada.
  5. Ejercen como componentes de software de caracter reutilizable.

1.0.1 Un poco de terminología

- Hemos de distinguir 3 roles fundamentales cuando empleamos servicios web:

  1. Proveedor del servicio: Es quien implemente el servicio y lo hace accesible a través de internet.
  2. Solicitante del servicio: Cualquier cliente que necesite usar una funcionalidad prestada.

3. UDDI (Universal Description and Discovery).
   - Es una especificación técnica para publicar y buscar servicios web. Dicha especificación define como construir un repositorio distribuido de servicios web.
   - Sirve para crear un registro en el que se publica información sobre servicios web: Que organizaciones los proporcionan, para que es cada servicio y cómo se pueden usar.
   - Los datos de este registro son almacenados en XML, pero el uso de los servicios a través de sus interfaces hace uso de una variante de este conocida como WSDL (Web Services Description Languaje).

1.0.2 Conceptos técnicos

- Los servicios web se apoyan en protocolos de comunicación estándar como HTTP, SMTP, FTP, etc.
- Algunos de los servicios web más conocidos (y que se apoyan sobre HTTP) son:
  - XML-RCP: Es un protocolo muy simple, codifica los datos en XML, solo define unos pocos tipos de datos y comandos útiles.
  - SOAP: Es un protocolo que resulta de la ampliación y mejora de XML-RCP por parte de Microsoft, aunque es menos simle que el XML-RCP original. Este hace uso de XML para coificar la información, entre la que se encuentra la operación de la que se quiere hacer uso y los parámetros de esta.
  - REST: Hace uso de los diferentes tipos de peticiones HTTP (GET,POST,PUT,DELETE) para especificar que operación se va a realizar. Dichas operaciones se realizan sobre recursos, cada uno de los cuales es identificado por una URL. A diferencia de los protocolos anteriores, este codifica la información en JSON.

1.0.3 Un enfoque práctico:

A partir de esta sección trataremos de esclarecer un poco más los conceptos anteriores. Para ello supondremos que somos un programadxr de una empresa que trabaja con Python 3 y que desea emplear el ERP-CRM de Odoo 13 que se ejecuta en un servidor como un almacén de datos de su empresa.

Conexión al servicio web

[1]:
```
import xmlrpc.client # Protocolo para hacer uso del servicio web (XML-RPC).

url = 'http://localhost:8069' # Dirección web y puerto del servidor Odoo 13.

try:

    # Intentamos establecer una onexión con el servicio web.
    # Accediendo a esta dirección solamente podemos pedir información o␣
 ↪autentificarnos.
    server = xmlrpc.client.ServerProxy('{}/xmlrpc/2/common'.format(url))
```

```
    print("La conexión se realizaó correctamente a", url)
    print("Datos del servidor:")

    # Información del servidor.
    for x, y in server.version().items():

        print(x,"-->",y)

except Exception as e:

    print("Error:",e);
```

```
La conexión se realizaó correctamente a http://localhost:8069
Datos del servidor:
server_version --> 13.0-20200224
server_version_info --> [13, 0, 0, 'final', 0, '']
server_serie --> 13.0
protocol_version --> 1
```

Con la cadena de conexión anterior de ServerProxy solo podremos pedir información al servidor o intentar registrarnos como un usuario, ya que invocar estas operaciones no requieren privilegios de ninguna clase. Pero si queremos invocar otras operaciones habremos de autentificarnos ante el servicio web.

[2]:
```
db = 'dreambeer' # Base de datos de Odoo a la que queremos acceder.
password = '12345678' # Contraseña de un usuario valido.
username = 'dreambeer@gmail.com' # Usuario autorizado.

# Nos identificamos ante el servidor que implementa el servicio, si la␣
 ↪autentificación es correcta entonces se nos
# asignará un identificador de usuario.
userId = server.authenticate(db, username, password, {})

if userId > 0:

    print("Identificador de usuario:", userId)

else:

    print("Usuario y/o contraseña incorrectos")
```

```
Identificador de usuario: 2
```

Debemos establecer una conexión a la siguiente URL si deseamos invocar los métodos remotos del servicio.

[3]:
```
server = xmlrpc.client.ServerProxy('{}/xmlrpc/2/object'.format(url))
```

Para invocar métodos remotos emplearemos la función **execute_kw**, esta tiene los siguientes parámetros:

- El nombre de una BDD (string).
- El id del usuario identificado (integer).
- La contraseña del usuario identificado (string).
- Nombre del modelo de datos al que conectarnos (string).
- Nombre del método que queremos invocar (string).
- Un array de parámetros que son pasados como argumento a la función invocada, el orden de estos importa.
- Un diccionario de claves y valores con parámetros opcionales.

Para consultar que permisos tenemos sobre el modelo de datos de los clientes (res.partner) invocaremos el método remoto **check_access_rights**, también indicaremos que los permisos se indiquen de forma lógica, que no levantando una excepción cuando alguno no se posea.

En la siguiente página podemos ver gran parte de las tablas del modelo de Odoo: OdooModel

[4]:
```
for right in ['create','write','read','unlink']:

    print(
        server.execute_kw(
            db, userId, password,
            'res.partner', 'check_access_rights',
            [right], {'raise_exception': False}
        )
    )
```

```
True
True
True
True
```

El método **search** devuelve todos los identificadores de los registros que coincidan con las condiciones que se pasan como parámetro.

[5]:
```
print(
    server.execute_kw(
        db, userId, password,
        'res.partner', 'search',
        [
            [
                ['is_company', '=', True],
                ['name','=','Bar Escocia la Verde']
            ]
        ]
    )
)
```

```
[15]
```

Por defecto una búsqueda devuelve todos los identificadores de los registros que cumplan con la condición de búsqueda. Sin embargo hay veces que queremos limitar el número de resultados, para ello utilizaremos los parámetros opcionales **offset** (que indica a partir de que registro de han de comenzar a mostrar resultados) y **limit** (que indica cuantos registros se puden mostrar como máximo).

[6]:
```
print(
    server.execute_kw(
        db, userId, password,
        'res.partner', 'search',
        [
            [
                ['is_company', '=', True]
            ]
        ],
        {
            'offset': 0,
            'limit': 5
        }
    )
)
```

```
[12, 15, 11, 20, 7]
```

Otra posible operación que se puede utilizar es **search_count**, esta es una variante de search que devuelve el número de registros que cumplieron la condición especificada.

[7]:
```
print(
    server.execute_kw(
        db, userId, password,
        'res.partner', 'search_count',
        [
            [
                ['is_company', '=', True]
            ]
        ]
    )
)
```

```
12
```

1.0.4 Operaciones CRUD

Si lo que queremos es leer todos los campos de los registros, hemos de combinar el método **search** con el método **read**, pues el primero devuelve todos los identificadores de los registros que nos interesan y el segundo extrae la información de dichos registros.

[8]:
```
import pprint as pp

ids = server.execute_kw(
            db, userId, password,
            'res.partner', 'search',
            [
                [
                    ['is_company', '=', True]
                ]
            ],
            {'limit': 4}
        )

print("Identificadores de las empresas: ", ids)

results = server.execute_kw(
                db, userId, password,
                'res.partner', 'read', [ids]
            )

for r in results:
    pp.pprint(r)
   ␣
 ↪print("**********************************************************************")
```

```
Identificadores de las empresas:  [12, 15, 11, 20]
{'__last_update': '2020-02-28 14:36:31',
 'active': True,
 'active_lang_count': 1,
 'activity_date_deadline': False,
 'activity_exception_decoration': False,
 'activity_exception_icon': False,
 'activity_ids': [],
 'activity_state': False,
 'activity_summary': False,
 'activity_type_id': False,
 'activity_user_id': False,
 'additional_info': False,
 'bank_account_count': 0,
 'bank_ids': [],
 'calendar_last_notif_ack': '2020-02-28 14:36:31',
 'can_publish': True,
 'category_id': [],
 'channel_ids': [],
 'child_ids': [],
 'city': 'Madrid',
 'color': 0,
```

```
'comment': False,
'commercial_company_name': 'Bar Cohete Soyuz',
'commercial_partner_id': [12, 'Bar Cohete Soyuz'],
'company_id': False,
'company_name': False,
'company_type': 'company',
'contact_address': 'Bar Cohete Soyuz\n\n\n Madrid\nSpain',
'contract_ids': [],
'country_id': [68, 'Spain'],
'create_date': '2020-02-28 14:36:31',
'create_uid': [2, 'Administrator'],
'credit': 0.0,
'credit_limit': 0.0,
'currency_id': [1, 'EUR'],
'customer_rank': 1,
'date': False,
'debit': 0.0,
'debit_limit': 0.0,
'display_name': 'Bar Cohete Soyuz',
'email': False,
'email_formatted': '',
'email_normalized': False,
'employee': False,
'function': False,
'has_unreconciled_entries': False,
'id': 12,
'im_status': 'im_partner',
'image_1024': False,
'image_128': False,
'image_1920': False,
'image_256': False,
'image_512': False,
'industry_id': False,
'invoice_ids': [],
'invoice_warn': 'no-message',
'invoice_warn_msg': False,
'is_blacklisted': False,
'is_company': True,
'is_published': False,
'journal_item_count': 0,
'lang': 'en_US',
'last_time_entries_checked': False,
'last_website_so_id': False,
'meeting_count': 0,
'meeting_ids': [],
'message_attachment_count': 0,
'message_bounce': 0,
'message_channel_ids': [],
```

```
'message_follower_ids': [16],
'message_has_error': False,
'message_has_error_counter': 0,
'message_has_sms_error': False,
'message_ids': [12],
'message_is_follower': True,
'message_main_attachment_id': False,
'message_needaction': False,
'message_needaction_counter': 0,
'message_partner_ids': [3],
'message_unread': False,
'message_unread_counter': 0,
'mobile': False,
'name': 'Bar Cohete Soyuz',
'opportunity_count': 1,
'opportunity_ids': [6],
'parent_id': False,
'parent_name': False,
'partner_gid': 0,
'partner_latitude': 0.0,
'partner_longitude': 0.0,
'partner_share': True,
'payment_token_count': 0,
'payment_token_ids': [],
'phone': False,
'phone_blacklisted': False,
'phone_sanitized': False,
'picking_warn': 'no-message',
'picking_warn_msg': False,
'property_account_payable_id': [188,
                                '410000 Acreedores por prestaciones de '
                                'servicios (euros)'],
'property_account_position_id': False,
'property_account_receivable_id': [193, '430000 Clientes (euros)'],
'property_payment_term_id': False,
'property_product_pricelist': [1, 'Public Pricelist (EUR)'],
'property_purchase_currency_id': False,
'property_stock_customer': [5, 'Partner Locations/Customers'],
'property_stock_supplier': [4, 'Partner Locations/Vendors'],
'property_supplier_payment_term_id': False,
'purchase_order_count': 0,
'purchase_warn': 'no-message',
'purchase_warn_msg': False,
'ref': False,
'ref_company_ids': [],
'sale_order_count': 0,
'sale_order_ids': [],
'sale_warn': 'no-message',
```

```
 'sale_warn_msg': False,
 'same_vat_partner_id': False,
 'self': [12, 'Bar Cohete Soyuz'],
 'signup_expiration': False,
 'signup_token': False,
 'signup_type': False,
 'signup_url': False,
 'signup_valid': False,
 'state_id': [448, 'Madrid (ES)'],
 'street': False,
 'street2': False,
 'supplier_invoice_count': 0,
 'supplier_rank': 0,
 'team_id': False,
 'title': False,
 'total_invoiced': 0.0,
 'trust': 'normal',
 'type': 'contact',
 'tz': False,
 'tz_offset': '+0000',
 'user_id': False,
 'user_ids': [],
 'vat': False,
 'visitor_ids': [],
 'website': False,
 'website_id': False,
 'website_message_ids': [],
 'website_published': False,
 'website_url': '#',
 'write_date': '2020-02-28 14:36:31',
 'write_uid': [2, 'Administrator'],
 'zip': False}
********************************************************************************
{'__last_update': '2020-02-28 14:40:31',
 'active': True,
 'active_lang_count': 1,
 'activity_date_deadline': False,
 'activity_exception_decoration': False,
 'activity_exception_icon': False,
 'activity_ids': [],
 'activity_state': False,
 'activity_summary': False,
 'activity_type_id': False,
 'activity_user_id': False,
 'additional_info': False,
 'bank_account_count': 0,
 'bank_ids': [],
 'calendar_last_notif_ack': '2020-02-28 14:40:31',
```

```
'can_publish': True,
'category_id': [],
'channel_ids': [],
'child_ids': [],
'city': 'Madrid',
'color': 0,
'comment': False,
'commercial_company_name': 'Bar Escocia la Verde',
'commercial_partner_id': [15, 'Bar Escocia la Verde'],
'company_id': False,
'company_name': False,
'company_type': 'company',
'contact_address': 'Bar Escocia la Verde\n\n\n Madrid\nSpain',
'contract_ids': [],
'country_id': [68, 'Spain'],
'create_date': '2020-02-28 14:40:31',
'create_uid': [2, 'Administrator'],
'credit': 0.0,
'credit_limit': 0.0,
'currency_id': [1, 'EUR'],
'customer_rank': 2,
'date': False,
'debit': 0.0,
'debit_limit': 0.0,
'display_name': 'Bar Escocia la Verde',
'email': False,
'email_formatted': '',
'email_normalized': False,
'employee': False,
'function': False,
'has_unreconciled_entries': False,
'id': 15,
'im_status': 'im_partner',
'image_1024': False,
'image_128': False,
'image_1920': False,
'image_256': False,
'image_512': False,
'industry_id': False,
'invoice_ids': [13, 14],
'invoice_warn': 'no-message',
'invoice_warn_msg': False,
'is_blacklisted': False,
'is_company': True,
'is_published': False,
'journal_item_count': 8,
'lang': 'en_US',
'last_time_entries_checked': False,
```

```
'last_website_so_id': False,
'meeting_count': 0,
'meeting_ids': [],
'message_attachment_count': 0,
'message_bounce': 0,
'message_channel_ids': [],
'message_follower_ids': [19],
'message_has_error': False,
'message_has_error_counter': 0,
'message_has_sms_error': False,
'message_ids': [15],
'message_is_follower': True,
'message_main_attachment_id': False,
'message_needaction': False,
'message_needaction_counter': 0,
'message_partner_ids': [3],
'message_unread': False,
'message_unread_counter': 0,
'mobile': False,
'name': 'Bar Escocia la Verde',
'opportunity_count': 1,
'opportunity_ids': [7],
'parent_id': False,
'parent_name': False,
'partner_gid': 0,
'partner_latitude': 0.0,
'partner_longitude': 0.0,
'partner_share': True,
'payment_token_count': 0,
'payment_token_ids': [],
'phone': False,
'phone_blacklisted': False,
'phone_sanitized': False,
'picking_warn': 'no-message',
'picking_warn_msg': False,
'property_account_payable_id': [188,
                                '410000 Acreedores por prestaciones de '
                                'servicios (euros)'],
'property_account_position_id': False,
'property_account_receivable_id': [193, '430000 Clientes (euros)'],
'property_payment_term_id': False,
'property_product_pricelist': [1, 'Public Pricelist (EUR)'],
'property_purchase_currency_id': False,
'property_stock_customer': [5, 'Partner Locations/Customers'],
'property_stock_supplier': [4, 'Partner Locations/Vendors'],
'property_supplier_payment_term_id': False,
'purchase_order_count': 0,
'purchase_warn': 'no-message',
```

```
 'purchase_warn_msg': False,
 'ref': False,
 'ref_company_ids': [],
 'sale_order_count': 1,
 'sale_order_ids': [1],
 'sale_warn': 'no-message',
 'sale_warn_msg': False,
 'same_vat_partner_id': False,
 'self': [15, 'Bar Escocia la Verde'],
 'signup_expiration': False,
 'signup_token': False,
 'signup_type': False,
 'signup_url': False,
 'signup_valid': False,
 'state_id': [448, 'Madrid (ES)'],
 'street': False,
 'street2': False,
 'supplier_invoice_count': 0,
 'supplier_rank': 0,
 'team_id': False,
 'title': False,
 'total_invoiced': 1200.0,
 'trust': 'normal',
 'type': 'contact',
 'tz': False,
 'tz_offset': '+0000',
 'user_id': False,
 'user_ids': [],
 'vat': False,
 'visitor_ids': [],
 'website': False,
 'website_id': False,
 'website_message_ids': [],
 'website_published': False,
 'website_url': '#',
 'write_date': '2020-02-28 14:40:31',
 'write_uid': [2, 'Administrator'],
 'zip': False}
********************************************************************************
{'__last_update': '2020-02-28 14:35:24',
 'active': True,
 'active_lang_count': 1,
 'activity_date_deadline': False,
 'activity_exception_decoration': False,
 'activity_exception_icon': False,
 'activity_ids': [],
 'activity_state': False,
 'activity_summary': False,
```

```
'activity_type_id': False,
'activity_user_id': False,
'additional_info': False,
'bank_account_count': 0,
'bank_ids': [],
'calendar_last_notif_ack': '2020-02-28 14:35:24',
'can_publish': True,
'category_id': [],
'channel_ids': [],
'child_ids': [],
'city': 'Madrid',
'color': 0,
'comment': False,
'commercial_company_name': 'Bar la Patatera',
'commercial_partner_id': [11, 'Bar la Patatera'],
'company_id': False,
'company_name': False,
'company_type': 'company',
'contact_address': 'Bar la Patatera\n\n\n Madrid\nSpain',
'contract_ids': [],
'country_id': [68, 'Spain'],
'create_date': '2020-02-28 14:35:24',
'create_uid': [2, 'Administrator'],
'credit': 0.0,
'credit_limit': 0.0,
'currency_id': [1, 'EUR'],
'customer_rank': 1,
'date': False,
'debit': 0.0,
'debit_limit': 0.0,
'display_name': 'Bar la Patatera',
'email': False,
'email_formatted': '',
'email_normalized': False,
'employee': False,
'function': False,
'has_unreconciled_entries': False,
'id': 11,
'im_status': 'im_partner',
'image_1024': False,
'image_128': False,
'image_1920': False,
'image_256': False,
'image_512': False,
'industry_id': False,
'invoice_ids': [],
'invoice_warn': 'no-message',
'invoice_warn_msg': False,
```

```
'is_blacklisted': False,
'is_company': True,
'is_published': False,
'journal_item_count': 0,
'lang': 'en_US',
'last_time_entries_checked': False,
'last_website_so_id': False,
'meeting_count': 0,
'meeting_ids': [],
'message_attachment_count': 0,
'message_bounce': 0,
'message_channel_ids': [],
'message_follower_ids': [15],
'message_has_error': False,
'message_has_error_counter': 0,
'message_has_sms_error': False,
'message_ids': [11],
'message_is_follower': True,
'message_main_attachment_id': False,
'message_needaction': False,
'message_needaction_counter': 0,
'message_partner_ids': [3],
'message_unread': False,
'message_unread_counter': 0,
'mobile': False,
'name': 'Bar la Patatera',
'opportunity_count': 1,
'opportunity_ids': [10],
'parent_id': False,
'parent_name': False,
'partner_gid': 0,
'partner_latitude': 0.0,
'partner_longitude': 0.0,
'partner_share': True,
'payment_token_count': 0,
'payment_token_ids': [],
'phone': False,
'phone_blacklisted': False,
'phone_sanitized': False,
'picking_warn': 'no-message',
'picking_warn_msg': False,
'property_account_payable_id': [188,
                                '410000 Acreedores por prestaciones de '
                                'servicios (euros)'],
'property_account_position_id': False,
'property_account_receivable_id': [193, '430000 Clientes (euros)'],
'property_payment_term_id': False,
'property_product_pricelist': [1, 'Public Pricelist (EUR)'],
```

```
 'property_purchase_currency_id': False,
 'property_stock_customer': [5, 'Partner Locations/Customers'],
 'property_stock_supplier': [4, 'Partner Locations/Vendors'],
 'property_supplier_payment_term_id': False,
 'purchase_order_count': 0,
 'purchase_warn': 'no-message',
 'purchase_warn_msg': False,
 'ref': False,
 'ref_company_ids': [],
 'sale_order_count': 0,
 'sale_order_ids': [],
 'sale_warn': 'no-message',
 'sale_warn_msg': False,
 'same_vat_partner_id': False,
 'self': [11, 'Bar la Patatera'],
 'signup_expiration': False,
 'signup_token': False,
 'signup_type': False,
 'signup_url': False,
 'signup_valid': False,
 'state_id': [448, 'Madrid (ES)'],
 'street': False,
 'street2': False,
 'supplier_invoice_count': 0,
 'supplier_rank': 0,
 'team_id': False,
 'title': False,
 'total_invoiced': 0.0,
 'trust': 'normal',
 'type': 'contact',
 'tz': False,
 'tz_offset': '+0000',
 'user_id': False,
 'user_ids': [],
 'vat': False,
 'visitor_ids': [],
 'website': False,
 'website_id': False,
 'website_message_ids': [],
 'website_published': False,
 'website_url': '#',
 'write_date': '2020-02-28 14:35:24',
 'write_uid': [2, 'Administrator'],
 'zip': False}
**************************************************************************
{'__last_update': '2020-03-17 19:40:38',
 'active': True,
 'active_lang_count': 1,
```

```
'activity_date_deadline': False,
'activity_exception_decoration': False,
'activity_exception_icon': False,
'activity_ids': [],
'activity_state': False,
'activity_summary': False,
'activity_type_id': False,
'activity_user_id': False,
'additional_info': False,
'bank_account_count': 0,
'bank_ids': [],
'calendar_last_notif_ack': '2020-03-17 19:40:38',
'can_publish': True,
'category_id': [],
'channel_ids': [],
'child_ids': [21],
'city': 'Yokdzonot',
'color': 0,
'comment': False,
'commercial_company_name': 'Chiringuito Playa Dorada',
'commercial_partner_id': [20, 'Chiringuito Playa Dorada'],
'company_id': False,
'company_name': False,
'company_type': 'company',
'contact_address': 'Chiringuito Playa Dorada\n'
                   'Playa de Akumal \n'
                   ' Yokdzonot, ROO\n'
                   'Mexico',
'contract_ids': [],
'country_id': [156, 'Mexico'],
'create_date': '2020-03-17 19:40:38',
'create_uid': [2, 'Administrator'],
'credit': 0.0,
'credit_limit': 0.0,
'currency_id': [1, 'EUR'],
'customer_rank': 0,
'date': False,
'debit': 0.0,
'debit_limit': 0.0,
'display_name': 'Chiringuito Playa Dorada',
'email': 'edbega@gmail.com',
'email_formatted': '"Chiringuito Playa Dorada" <edbega@gmail.com>',
'email_normalized': 'edbega@gmail.com',
'employee': False,
'function': 'Manager',
'has_unreconciled_entries': False,
'id': 20,
'im_status': 'im_partner',
```

```
'image_1024': False,
'image_128': False,
'image_1920': False,
'image_256': False,
'image_512': False,
'industry_id': False,
'invoice_ids': [21, 17],
'invoice_warn': 'no-message',
'invoice_warn_msg': False,
'is_blacklisted': False,
'is_company': True,
'is_published': False,
'journal_item_count': 6,
'lang': 'en_US',
'last_time_entries_checked': False,
'last_website_so_id': False,
'meeting_count': 0,
'meeting_ids': [],
'message_attachment_count': 0,
'message_bounce': 0,
'message_channel_ids': [],
'message_follower_ids': [248, 249],
'message_has_error': False,
'message_has_error_counter': 0,
'message_has_sms_error': False,
'message_ids': [500],
'message_is_follower': True,
'message_main_attachment_id': False,
'message_needaction': False,
'message_needaction_counter': 0,
'message_partner_ids': [3, 18],
'message_unread': False,
'message_unread_counter': 0,
'mobile': False,
'name': 'Chiringuito Playa Dorada',
'opportunity_count': 2,
'opportunity_ids': [],
'parent_id': False,
'parent_name': False,
'partner_gid': 0,
'partner_latitude': 0.0,
'partner_longitude': 0.0,
'partner_share': True,
'payment_token_count': 0,
'payment_token_ids': [],
'phone': False,
'phone_blacklisted': False,
'phone_sanitized': False,
```

```
'picking_warn': 'no-message',
'picking_warn_msg': False,
'property_account_payable_id': [188,
                                '410000 Acreedores por prestaciones de '
                                'servicios (euros)'],
'property_account_position_id': False,
'property_account_receivable_id': [193, '430000 Clientes (euros)'],
'property_payment_term_id': False,
'property_product_pricelist': [1, 'Public Pricelist (EUR)'],
'property_purchase_currency_id': False,
'property_stock_customer': [5, 'Partner Locations/Customers'],
'property_stock_supplier': [4, 'Partner Locations/Vendors'],
'property_supplier_payment_term_id': False,
'purchase_order_count': 0,
'purchase_warn': 'no-message',
'purchase_warn_msg': False,
'ref': False,
'ref_company_ids': [],
'sale_order_count': 2,
'sale_order_ids': [],
'sale_warn': 'no-message',
'sale_warn_msg': False,
'same_vat_partner_id': False,
'self': [20, 'Chiringuito Playa Dorada'],
'signup_expiration': False,
'signup_token': False,
'signup_type': False,
'signup_url': False,
'signup_valid': False,
'state_id': [506, 'Quintana Roo (MX)'],
'street': 'Playa de Akumal',
'street2': False,
'supplier_invoice_count': 0,
'supplier_rank': 0,
'team_id': [5, 'Latinoamérica'],
'title': [3, 'Mister'],
'total_invoiced': 2760.0,
'trust': 'normal',
'type': 'contact',
'tz': False,
'tz_offset': '+0000',
'user_id': [6, 'Pepe Pérez'],
'user_ids': [],
'vat': False,
'visitor_ids': [],
'website': 'http://www.barplayadorada.com',
'website_id': False,
'website_message_ids': [],
```

```
 'website_published': False,
 'website_url': '#',
 'write_date': '2020-03-17 19:40:38',
 'write_uid': [2, 'Administrator'],
 'zip': False}
**************************************************************************
```

Otra opción que tenemos con respecto a la lectura es indicar que columnas son las que queremos leer. Para poder indicar esta información haremos uso del diccionario de parámetros opcionales, usando el parámetro **fields**.

[9]:
```
ids = server.execute_kw(
            db, userId, password,
            'res.partner', 'search',
            [
                [
                    ['is_company', '=', True]
                ]
            ],
            {'limit': 4}
        )

print("Identificadores de las empresas: ", ids)

results = server.execute_kw(
                db, userId, password,
                'res.partner', 'read', [ids],
                {'fields': ['name', 'country_id', 'comment']}
            )

for r in results:
    pp.pprint(r)
  ␣
 ↪print("**************************************************************************")
```

```
Identificadores de las empresas:  [12, 15, 11, 20]
{'comment': False,
 'country_id': [68, 'Spain'],
 'id': 12,
 'name': 'Bar Cohete Soyuz'}
**************************************************************************
{'comment': False,
 'country_id': [68, 'Spain'],
 'id': 15,
 'name': 'Bar Escocia la Verde'}
**************************************************************************
{'comment': False,
 'country_id': [68, 'Spain'],
```

```
 'id': 11,
 'name': 'Bar la Patatera'}
**************************************************************************
{'comment': False,
 'country_id': [156, 'Mexico'],
 'id': 20,
 'name': 'Chiringuito Playa Dorada'}
**************************************************************************
```

Si lo que queremos es extraer información de determinados atributos de un modelo de datos, podemos recurrir al parámetro opcional **attributes**, este permite especificar que información queremos extraer de cada atributo. La información más habitual es:

- La etiqueta del campo (string).
- Un texto de ayuda sobre el campo (help).
- El tipo de valor del campo (type).

[10]:
```
result = server.execute_kw(
    db, userId, password, 'res.partner', 'fields_get',
    [], {'attributes': ['string', 'help', 'type']}
)

pp.pprint(result);
```

```
{'__last_update': {'string': 'Last Modified on', 'type': 'datetime'},
 'active': {'string': 'Active', 'type': 'boolean'},
 'active_lang_count': {'string': 'Active Lang Count', 'type': 'integer'},
 'activity_date_deadline': {'string': 'Next Activity Deadline', 'type': 'date'},
 'activity_exception_decoration': {'help': 'Type of the exception activity on '
                                           'record.',
                                   'string': 'Activity Exception Decoration',
                                   'type': 'selection'},
 'activity_exception_icon': {'help': 'Icon to indicate an exception activity.',
                             'string': 'Icon',
                             'type': 'char'},
 'activity_ids': {'string': 'Activities', 'type': 'one2many'},
 'activity_state': {'help': 'Status based on activities\n'
                            'Overdue: Due date is already passed\n'
                            'Today: Activity date is today\n'
                            'Planned: Future activities.',
                    'string': 'Activity State',
                    'type': 'selection'},
 'activity_summary': {'string': 'Next Activity Summary', 'type': 'char'},
 'activity_type_id': {'string': 'Next Activity Type', 'type': 'many2one'},
 'activity_user_id': {'string': 'Responsible User', 'type': 'many2one'},
 'additional_info': {'string': 'Additional info', 'type': 'char'},
 'bank_account_count': {'string': 'Bank', 'type': 'integer'},
 'bank_ids': {'string': 'Banks', 'type': 'one2many'},
 'calendar_last_notif_ack': {'string': 'Last notification marked as read from '
```

```
                                    'base Calendar',
                             'type': 'datetime'},
'can_publish': {'string': 'Can Publish', 'type': 'boolean'},
'category_id': {'string': 'Tags', 'type': 'many2many'},
'channel_ids': {'string': 'Channels', 'type': 'many2many'},
'child_ids': {'string': 'Contact', 'type': 'one2many'},
'city': {'string': 'City', 'type': 'char'},
'color': {'string': 'Color Index', 'type': 'integer'},
'comment': {'string': 'Notes', 'type': 'text'},
'commercial_company_name': {'string': 'Company Name Entity', 'type': 'char'},
'commercial_partner_id': {'string': 'Commercial Entity', 'type': 'many2one'},
'company_id': {'string': 'Company', 'type': 'many2one'},
'company_name': {'string': 'Company Name', 'type': 'char'},
'company_type': {'string': 'Company Type', 'type': 'selection'},
'contact_address': {'string': 'Complete Address', 'type': 'char'},
'contract_ids': {'string': 'Partner Contracts', 'type': 'one2many'},
'country_id': {'string': 'Country', 'type': 'many2one'},
'create_date': {'string': 'Created on', 'type': 'datetime'},
'create_uid': {'string': 'Created by', 'type': 'many2one'},
'credit': {'help': 'Total amount this customer owes you.',
           'string': 'Total Receivable',
           'type': 'monetary'},
'credit_limit': {'string': 'Credit Limit', 'type': 'float'},
'currency_id': {'help': 'Utility field to express amount currency',
                'string': 'Currency',
                'type': 'many2one'},
'customer_rank': {'string': 'Customer Rank', 'type': 'integer'},
'date': {'string': 'Date', 'type': 'date'},
'debit': {'help': 'Total amount you have to pay to this vendor.',
          'string': 'Total Payable',
          'type': 'monetary'},
'debit_limit': {'string': 'Payable Limit', 'type': 'monetary'},
'display_name': {'string': 'Display Name', 'type': 'char'},
'email': {'string': 'Email', 'type': 'char'},
'email_formatted': {'help': 'Format email address "Name <email@domain>"',
                    'string': 'Formatted Email',
                    'type': 'char'},
'email_normalized': {'help': 'This field is used to search on email address '
                             'as the primary email field can contain more '
                             'than strictly an email address.',
                     'string': 'Normalized Email',
                     'type': 'char'},
'employee': {'help': 'Check this box if this contact is an Employee.',
             'string': 'Employee',
             'type': 'boolean'},
'function': {'string': 'Job Position', 'type': 'char'},
'has_unreconciled_entries': {'help': 'The partner has at least one '
                                     'unreconciled debit and credit since '
```

```
                                    'last time the invoices & payments '
                                    'matching was performed.',
                           'string': 'Has Unreconciled Entries',
                           'type': 'boolean'},
'id': {'string': 'ID', 'type': 'integer'},
'im_status': {'string': 'IM Status', 'type': 'char'},
'image_1024': {'string': 'Image 1024', 'type': 'binary'},
'image_128': {'string': 'Image 128', 'type': 'binary'},
'image_1920': {'string': 'Image', 'type': 'binary'},
'image_256': {'string': 'Image 256', 'type': 'binary'},
'image_512': {'string': 'Image 512', 'type': 'binary'},
'industry_id': {'string': 'Industry', 'type': 'many2one'},
'invoice_ids': {'string': 'Invoices', 'type': 'one2many'},
'invoice_warn': {'help': 'Selecting the "Warning" option will notify user '
                         'with the message, Selecting "Blocking Message" will '
                         'throw an exception with the message and block the '
                         'flow. The Message has to be written in the next '
                         'field.',
                 'string': 'Invoice',
                 'type': 'selection'},
'invoice_warn_msg': {'string': 'Message for Invoice', 'type': 'text'},
'is_blacklisted': {'help': 'If the email address is on the blacklist, the '
                           "contact won't receive mass mailing anymore, from "
                           'any list',
                   'string': 'Blacklist',
                   'type': 'boolean'},
'is_company': {'help': 'Check if the contact is a company, otherwise it is a '
                       'person',
               'string': 'Is a Company',
               'type': 'boolean'},
'is_published': {'string': 'Is Published', 'type': 'boolean'},
'journal_item_count': {'string': 'Journal Items', 'type': 'integer'},
'lang': {'help': 'All the emails and documents sent to this contact will be '
                 'translated in this language.',
         'string': 'Language',
         'type': 'selection'},
'last_time_entries_checked': {'help': 'Last time the invoices & payments '
                                      'matching was performed for this '
                                      "partner. It is set either if there's "
                                      'not at least an unreconciled debit and '
                                      'an unreconciled credit or if you click '
                                      'the "Done" button.',
                              'string': 'Latest Invoices & Payments Matching '
                                        'Date',
                              'type': 'datetime'},
'last_website_so_id': {'string': 'Last Online Sales Order',
                       'type': 'many2one'},
'meeting_count': {'string': '# Meetings', 'type': 'integer'},
```

```
'meeting_ids': {'string': 'Meetings', 'type': 'many2many'},
'message_attachment_count': {'string': 'Attachment Count', 'type': 'integer'},
'message_bounce': {'help': 'Counter of the number of bounced emails for this '
                           'contact',
                   'string': 'Bounce',
                   'type': 'integer'},
'message_channel_ids': {'string': 'Followers (Channels)', 'type': 'many2many'},
'message_follower_ids': {'string': 'Followers', 'type': 'one2many'},
'message_has_error': {'help': 'If checked, some messages have a delivery '
                              'error.',
                      'string': 'Message Delivery error',
                      'type': 'boolean'},
'message_has_error_counter': {'help': 'Number of messages with delivery error',
                              'string': 'Number of errors',
                              'type': 'integer'},
'message_has_sms_error': {'help': 'If checked, some messages have a delivery '
                                  'error.',
                          'string': 'SMS Delivery error',
                          'type': 'boolean'},
'message_ids': {'string': 'Messages', 'type': 'one2many'},
'message_is_follower': {'string': 'Is Follower', 'type': 'boolean'},
'message_main_attachment_id': {'string': 'Main Attachment',
                               'type': 'many2one'},
'message_needaction': {'help': 'If checked, new messages require your '
                               'attention.',
                       'string': 'Action Needed',
                       'type': 'boolean'},
'message_needaction_counter': {'help': 'Number of messages which requires an '
                                       'action',
                               'string': 'Number of Actions',
                               'type': 'integer'},
'message_partner_ids': {'string': 'Followers (Partners)', 'type': 'many2many'},
'message_unread': {'help': 'If checked, new messages require your attention.',
                   'string': 'Unread Messages',
                   'type': 'boolean'},
'message_unread_counter': {'help': 'Number of unread messages',
                           'string': 'Unread Messages Counter',
                           'type': 'integer'},
'mobile': {'string': 'Mobile', 'type': 'char'},
'name': {'string': 'Name', 'type': 'char'},
'opportunity_count': {'string': 'Opportunity', 'type': 'integer'},
'opportunity_ids': {'string': 'Opportunities', 'type': 'one2many'},
'parent_id': {'string': 'Related Company', 'type': 'many2one'},
'parent_name': {'string': 'Parent name', 'type': 'char'},
'partner_gid': {'string': 'Company database ID', 'type': 'integer'},
'partner_latitude': {'string': 'Geo Latitude', 'type': 'float'},
'partner_longitude': {'string': 'Geo Longitude', 'type': 'float'},
'partner_share': {'help': 'Either customer (not a user), either shared user. '
```

```
                        'Indicated the current partner is a customer '
                        'without access or with a limited access created '
                        'for sharing data.',
                'string': 'Share Partner',
                'type': 'boolean'},
'payment_token_count': {'string': 'Count Payment Token', 'type': 'integer'},
'payment_token_ids': {'string': 'Payment Tokens', 'type': 'one2many'},
'phone': {'string': 'Phone', 'type': 'char'},
'phone_blacklisted': {'help': 'If the email address is on the blacklist, the '
                              "contact won't receive mass mailing anymore, "
                              'from any list',
                      'string': 'Phone Blacklisted',
                      'type': 'boolean'},
'phone_sanitized': {'help': 'Field used to store sanitized phone number. '
                            'Helps speeding up searches and comparisons.',
                    'string': 'Sanitized Number',
                    'type': 'char'},
'picking_warn': {'help': 'Selecting the "Warning" option will notify user '
                         'with the message, Selecting "Blocking Message" will '
                         'throw an exception with the message and block the '
                         'flow. The Message has to be written in the next '
                         'field.',
                 'string': 'Stock Picking',
                 'type': 'selection'},
'picking_warn_msg': {'string': 'Message for Stock Picking', 'type': 'text'},
'property_account_payable_id': {'help': 'This account will be used instead of '
                                        'the default one as the payable '
                                        'account for the current partner',
                                'string': 'Account Payable',
                                'type': 'many2one'},
'property_account_position_id': {'help': 'The fiscal position determines the '
                                         'taxes/accounts used for this '
                                         'contact.',
                                 'string': 'Fiscal Position',
                                 'type': 'many2one'},
'property_account_receivable_id': {'help': 'This account will be used instead '
                                           'of the default one as the '
                                           'receivable account for the '
                                           'current partner',
                                   'string': 'Account Receivable',
                                   'type': 'many2one'},
'property_payment_term_id': {'help': 'This payment term will be used instead '
                                     'of the default one for sales orders and '
                                     'customer invoices',
                             'string': 'Customer Payment Terms',
                             'type': 'many2one'},
'property_product_pricelist': {'help': 'This pricelist will be used, instead '
                                       'of the default one, for sales to the '
```

```
                                       'current partner',
                                'string': 'Pricelist',
                                'type': 'many2one'},
'property_purchase_currency_id': {'help': 'This currency will be used, '
                                          'instead of the default one, for '
                                          'purchases from the current partner',
                                  'string': 'Supplier Currency',
                                  'type': 'many2one'},
'property_stock_customer': {'help': 'The stock location used as destination '
                                    'when sending goods to this contact.',
                            'string': 'Customer Location',
                            'type': 'many2one'},
'property_stock_supplier': {'help': 'The stock location used as source when '
                                    'receiving goods from this contact.',
                            'string': 'Vendor Location',
                            'type': 'many2one'},
'property_supplier_payment_term_id': {'help': 'This payment term will be used '
                                              'instead of the default one for '
                                              'purchase orders and vendor '
                                              'bills',
                                      'string': 'Vendor Payment Terms',
                                      'type': 'many2one'},
'purchase_order_count': {'string': 'Purchase Order Count', 'type': 'integer'},
'purchase_warn': {'help': 'Selecting the "Warning" option will notify user '
                          'with the message, Selecting "Blocking Message" '
                          'will throw an exception with the message and block '
                          'the flow. The Message has to be written in the '
                          'next field.',
                  'string': 'Purchase Order',
                  'type': 'selection'},
'purchase_warn_msg': {'string': 'Message for Purchase Order', 'type': 'text'},
'ref': {'string': 'Reference', 'type': 'char'},
'ref_company_ids': {'string': 'Companies that refers to partner',
                    'type': 'one2many'},
'sale_order_count': {'string': 'Sale Order Count', 'type': 'integer'},
'sale_order_ids': {'string': 'Sales Order', 'type': 'one2many'},
'sale_warn': {'help': 'Selecting the "Warning" option will notify user with '
                      'the message, Selecting "Blocking Message" will throw '
                      'an exception with the message and block the flow. The '
                      'Message has to be written in the next field.',
              'string': 'Sales Warnings',
              'type': 'selection'},
'sale_warn_msg': {'string': 'Message for Sales Order', 'type': 'text'},
'same_vat_partner_id': {'string': 'Partner with same Tax ID',
                        'type': 'many2one'},
'self': {'string': 'Self', 'type': 'many2one'},
'signup_expiration': {'string': 'Signup Expiration', 'type': 'datetime'},
'signup_token': {'string': 'Signup Token', 'type': 'char'},
```

```
'signup_type': {'string': 'Signup Token Type', 'type': 'char'},
'signup_url': {'string': 'Signup URL', 'type': 'char'},
'signup_valid': {'string': 'Signup Token is Valid', 'type': 'boolean'},
'state_id': {'string': 'State', 'type': 'many2one'},
'street': {'string': 'Street', 'type': 'char'},
'street2': {'string': 'Street2', 'type': 'char'},
'supplier_invoice_count': {'string': '# Vendor Bills', 'type': 'integer'},
'supplier_rank': {'string': 'Supplier Rank', 'type': 'integer'},
'team_id': {'help': 'If set, this Sales Team will be used for sales and '
                    'assignations related to this partner',
            'string': 'Sales Team',
            'type': 'many2one'},
'title': {'string': 'Title', 'type': 'many2one'},
'total_invoiced': {'string': 'Total Invoiced', 'type': 'monetary'},
'trust': {'string': 'Degree of trust you have in this debtor',
          'type': 'selection'},
'type': {'help': 'Invoice & Delivery addresses are used in sales orders. '
                 'Private addresses are only visible by authorized users.',
         'string': 'Address Type',
         'type': 'selection'},
'tz': {'help': 'When printing documents and exporting/importing data, time '
               'values are computed according to this timezone.\n'
               'If the timezone is not set, UTC (Coordinated Universal Time) '
               'is used.\n'
               'Anywhere else, time values are computed according to the time '
               'offset of your web client.',
       'string': 'Timezone',
       'type': 'selection'},
'tz_offset': {'string': 'Timezone offset', 'type': 'char'},
'user_id': {'help': 'The internal user in charge of this contact.',
            'string': 'Salesperson',
            'type': 'many2one'},
'user_ids': {'string': 'Users', 'type': 'one2many'},
'vat': {'help': 'The Tax Identification Number. Complete it if the contact is '
                'subjected to government taxes. Used in some legal '
                'statements.',
        'string': 'Tax ID',
        'type': 'char'},
'visitor_ids': {'string': 'Visitors', 'type': 'many2many'},
'website': {'string': 'Website Link', 'type': 'char'},
'website_id': {'help': 'Restrict publishing to this website.',
               'string': 'Website',
               'type': 'many2one'},
'website_message_ids': {'help': 'Website communication history',
                        'string': 'Website Messages',
                        'type': 'one2many'},
'website_published': {'string': 'Visible on current website',
                      'type': 'boolean'},
```

```
 'website_url': {'help': 'The full URL to access the document through the '
                         'website.',
                 'string': 'Website URL',
                 'type': 'char'},
 'write_date': {'string': 'Last Updated on', 'type': 'datetime'},
 'write_uid': {'string': 'Last Updated by', 'type': 'many2one'},
 'zip': {'string': 'Zip', 'type': 'char'}}
```

Como hemos visto anteriormente, tener que realizar la búsqueda de los id's con **search** para luego mostrar los datos de dichos registros mediante **read** es bastante incómodo. Esta es la razón por la cual se desarrolló el método **search_read** que equivale a la combinación de ambos. Los argumentos del método son como los del método **search**, pero soporta los parámetros opcionales del método **read**.

[11]:
```
results = server.execute_kw(
            db, userId, password,
            'res.partner', 'search_read',
            [[['is_company', '=', True]]],
            {'fields': ['name', 'country_id', 'comment'], 'limit': 5}
        )

for r in results:
    pp.pprint(r)
  ␣
 ↪print("**********************************************************************")
```

```
{'comment': False,
 'country_id': [68, 'Spain'],
 'id': 12,
 'name': 'Bar Cohete Soyuz'}
**********************************************************************
{'comment': False,
 'country_id': [68, 'Spain'],
 'id': 15,
 'name': 'Bar Escocia la Verde'}
**********************************************************************
{'comment': False,
 'country_id': [68, 'Spain'],
 'id': 11,
 'name': 'Bar la Patatera'}
**********************************************************************
{'comment': False,
 'country_id': [156, 'Mexico'],
 'id': 20,
 'name': 'Chiringuito Playa Dorada'}
**********************************************************************
{'comment': False,
 'country_id': [68, 'Spain'],
```

```
 'id': 7,
 'name': 'Chocolate Factory'}
**************************************************************************
```

Para insertar información en un modelo de datos hemos de cambiar la operación a **create** y a continuación especificar un array con los datos necesarios para crear el elemento que deseemos.

[12]:
```
insertId = server.execute_kw(db, userId, password, 'res.partner', 'create',␣
 ↪[{'name': "Example partner"}])

print("Se ha insertado el partner con el id:", insertId)
```

```
Se ha insertado el partner con el id: 62
```

Para modificar la información introducida se hace uso del método **write**, cuya sintaxis es igual a la de **create**, pero usándose para actualizar datos.

[13]:
```
server.execute_kw(db, userId, password, 'res.partner', 'write', [[insertId],␣
 ↪{'name': "New Example Partner"}])
```

[13]:
```
True
```

Ahora podemos llamar a un **getter** del atributo **name** de dicho **partner** para ver si realmente el registro quedó actualizado.

[14]:
```
server.execute_kw(db, userId, password, 'res.partner', 'name_get', [[insertId]])
```

[14]:
```
[[62, 'New Example Partner']]
```

Finalmente, para borrar registros disponemos del método **unlink**, este borrará todos los registros que cumplan la condición especificada.

[15]:
```
server.execute_kw(db, userId, password, 'res.partner', 'unlink', [[insertId]])

server.execute_kw(db, userId, password,'res.partner', 'search', [[['id', '=',␣
 ↪insertId]]])
```

[15]:
```
[]
```

1.0.5 Planteamiento de consultas con fines de Business Intelligence

---

La siguientes consultas se usarán con la finalidad de un sistema SCM (Supply Chain Management) para proveer datos a un Planning Manager y que así este pueda realizar los análisis convenientes para la realización de sus tareas dentro del SCP (Supply Chain Management):

1. Planificación de la demanda.

2. Planificación de lo que se nos ha de proveer.

3. Planificación de lo que se ha de tener en el almacén.

4. Planificación de lo que se habrá de producir.

5. Planificación de la logística para la distribución de productos.

- La siguiente consulta devuelve el total de unidades disponibles de productos, tanto de aquellos que se pueden vender a neustros clientes como aquellos que permanecen dentro de la empresa y que serán utilizados para realizar la fabricación del producto final.

[16]:
```
results = server.execute_kw(
            db, userId, password,
            'product.product', 'search_read',
            [],
            {'fields': ['name','qty_available']}
        )

data0 = dict();

for r in results:

    data0[r["name"]] = r["qty_available"];
    pp.pprint(r)
  ␣
 ↪print("**********************************************************************")
```

```
{'id': 16, 'name': 'botellas', 'qty_available': 13500.0}
**********************************************************************
{'id': 25, 'name': 'lupulo', 'qty_available': 6000.0}
**********************************************************************
{'id': 26, 'name': 'mango', 'qty_available': 1000.0}
**********************************************************************
{'id': 27, 'name': 'papaya', 'qty_available': 1300.0}
**********************************************************************
{'id': 28, 'name': 'piña', 'qty_available': 1200.0}
**********************************************************************
{'id': 29, 'name': 'trigo', 'qty_available': 13150.0}
**********************************************************************
{'id': 30, 'name': 'Tropical', 'qty_available': 29270.0}
**********************************************************************
{'id': 17, 'name': 'cebada', 'qty_available': 9050.0}
**********************************************************************
{'id': 18, 'name': 'chocolate', 'qty_available': 600.0}
**********************************************************************
{'id': 19, 'name': 'Coulant Cervecero', 'qty_available': 11800.0}
**********************************************************************
```

```
{'id': 20, 'name': 'etiqueta', 'qty_available': 7500.0}
**************************************************************************
{'id': 21, 'name': 'Exotic', 'qty_available': 26350.0}
**************************************************************************
{'id': 22, 'name': 'lata_cerveza', 'qty_available': 12000.0}
**************************************************************************
{'id': 23, 'name': 'Lemon Spirit', 'qty_available': 14799.0}
**************************************************************************
{'id': 24, 'name': 'limon', 'qty_available': 3000.0}
**************************************************************************
```

- La siguiente consulta extrae la cantidad de entradas de material (Input Transfers) que se han producido a lo largo del tiempo.

[17]:
```
results = server.execute_kw(
            db, userId, password
            , 'stock.move', 'search_read'
            , [[['reference','like','/IN/'],['description_picking','!=', '']]]
            , {'fields': ['description_picking','product_qty']}
        )

data1 = dict();

for r in results:
    data1[r["description_picking"]] = r["product_qty"];
    pp.pprint(r)
   ␣
 ↪print("**************************************************************************")
```

```
{'description_picking': 'chocolate', 'id': 193, 'product_qty': 600.0}
**************************************************************************
{'description_picking': 'chocolate', 'id': 195, 'product_qty': 600.0}
**************************************************************************
{'description_picking': 'chocolate', 'id': 198, 'product_qty': 600.0}
**************************************************************************
{'description_picking': 'limon', 'id': 231, 'product_qty': 3000.0}
**************************************************************************
{'description_picking': 'lata_cerveza', 'id': 233, 'product_qty': 12000.0}
**************************************************************************
{'description_picking': 'chocolate', 'id': 155, 'product_qty': 100.0}
**************************************************************************
{'description_picking': 'chocolate', 'id': 156, 'product_qty': 50.0}
**************************************************************************
{'description_picking': 'lata_cerveza', 'id': 157, 'product_qty': 1000.0}
**************************************************************************
{'description_picking': 'lata_cerveza', 'id': 158, 'product_qty': 2500.0}
**************************************************************************
{'description_picking': 'lata_cerveza', 'id': 159, 'product_qty': 500.0}
```

```
**************************************************************************
{'description_picking': 'limon', 'id': 160, 'product_qty': 800.0}
**************************************************************************
{'description_picking': 'trigo', 'id': 190, 'product_qty': 9000.0}
**************************************************************************
{'description_picking': 'cebada', 'id': 191, 'product_qty': 5000.0}
**************************************************************************
{'description_picking': 'coulant cervecero', 'id': 252, 'product_qty': 15000.0}
**************************************************************************
```

- La siguiente consulta extrae la cantidad de salida de productos para su venta (Output Transfers) que se han producido a lo largo del tiempo.

[18]:
```
results = server.execute_kw(
            db, userId, password
            , 'stock.move', 'search_read'
            , [[['reference','like','/OUT/'],['description_picking','!=', '']]]
            , {'fields': ['description_picking','product_qty']}
        )

data2 = dict();

for r in results:
    data2[r["description_picking"]] = r["product_qty"];
    pp.pprint(r)
  ␣
↪print("**************************************************************************")
```

```
{'description_picking': 'tropical', 'id': 234, 'product_qty': 200.0}
**************************************************************************
{'description_picking': 'coulant cervecero', 'id': 235, 'product_qty': 200.0}
**************************************************************************
{'description_picking': 'exotic', 'id': 236, 'product_qty': 200.0}
**************************************************************************
{'description_picking': 'lemon Spirit', 'id': 237, 'product_qty': 200.0}
**************************************************************************
{'description_picking': 'lemon Spirit', 'id': 238, 'product_qty': 200.0}
**************************************************************************
{'description_picking': 'coulant cervecero', 'id': 239, 'product_qty': 200.0}
**************************************************************************
{'description_picking': 'exotic', 'id': 240, 'product_qty': 200.0}
**************************************************************************
{'description_picking': 'tropical', 'id': 241, 'product_qty': 200.0}
**************************************************************************
{'description_picking': 'tropical', 'id': 242, 'product_qty': 600.0}
**************************************************************************
{'description_picking': 'coulant cervecero', 'id': 248, 'product_qty': 600.0}
**************************************************************************
```

```
{'description_picking': 'lemon Spirit', 'id': 249, 'product_qty': 600.0}
**********************************************************************
{'description_picking': 'coulant cervecero', 'id': 250, 'product_qty': 600.0}
**********************************************************************
{'description_picking': 'lemon Spirit', 'id': 251, 'product_qty': 600.0}
**********************************************************************
```

1.0.6 Uso de XAMPP con Python

A continuación se muestran unos ejemplos de como introducir y consultar los datos obtenidos en las queries vistas anteriormente en una BDD relacional MySQL gracias un servidor XAMPP instalado de forma local.

1. Conexión.

[19]:
```
import pymysql

# Abrimos una conexión a la base de datos.
db = pymysql.connect("localhost","root","","Dreambeer")

# Creamos un objeto cursos para poder trabajar con la base de datos.
cursor = db.cursor();
```

2. Consulta del stock disponible.

[20]:
```
botellas = data0["botellas"] if ("botellas" in data0) else 0
cebada = data0["cebada"] if ("cebada" in data0) else 0
chocolate = data0["chocolate"] if ("chocolate" in data0) else 0
etiqueta = data0["etiqueta"] if ("etiqueta" in data0) else 0
lata_cerveza = data0["lata_cerveza"] if ("lata_cerveza" in data0) else 0
limon = data0["limon"] if ("limon" in data0) else 0
lupulo = data0["lupulo"] if ("lupulo" in data0) else 0
mango = data0["mango"] if ("mango" in data0) else 0
papaya = data0["papaya"] if ("papaya" in data0) else 0
piña = data0["piña"] if ("piña" in data0) else 0
trigo = data0["trigo"] if ("trigo" in data0) else 0

colulantCervecero = data0["Coulant Cervecero"] if ("Coulant Cervecero" in␣
 ↪data0) else 0
exotic = data0["Exotic"] if ("Exotic" in data0) else 0
lemonSpirit = data0["Lemon Spirit"] if ("Lemon Spirit" in data0) else 0
tropical = data0["Tropical"] if ("Tropical" in data0) else 0

# La clave primaria no se indica en el insert ya que es el timestamp actual.
cursor.execute(
    "INSERT INTO " +
```

```
    "`Stock`(" +
        "`botellas`,`cebada`,`chocolate`,`etiqueta`,`lata_cerveza`," +
        "`limon`,`lupulo`,`mango`,`papaya`,`piña`,`trigo`," +
        "`coulant cervecero`,`exotic`,`lemon Spirit`,`tropical`" +
    ") " +
    "VALUES({},{},{},{},{},{},{},{},{},{},{},{},{},{},{})".format(
        botellas, cebada, chocolate, etiqueta, lata_cerveza,
        limon, lupulo, mango, papaya, piña, trigo,
        colulantCervecero, exotic, lemonSpirit, tropical
    )
)

# Confirmamos la inserción.
db.commit()

# Seleccionamos todos los registros de la tabla.
cursor.execute("SELECT * FROM `Stock`")

# Creamos un objeto iterador para que se mueva sobre el resultado.
rows = cursor.fetchall();

print(
    "botellas - cebada - chocolate - etiqueta - lata_cerveza - " +
    "limon - lupulo - mango - papaya - piña - trigo - " +
    "coulant cervecero - exotic - lemon Spirit - tropical\n"
)

# Mostramos todos los registros.
for row in rows:
    ␣
 ↪print("********************************************************************************")
    print(row[0], row[slice(1,len(row))])
```

```
botellas - cebada - chocolate - etiqueta - lata_cerveza - limon - lupulo - mango
- papaya - piña - trigo - coulant cervecero - exotic - lemon Spirit - tropical

********************************************************************************
2020-05-19 23:51:38 (13500, 9050, 600, 7500, 12000, 3000, 6000, 1000, 1300,
1200, 13150, 11800, 26350, 14799, 29270)
```

3. Consulta de las entradas de productos (Input Transfers).

[21]:
```
botellas = data1["botellas"] if ("botellas" in data1) else 0
cebada = data1["cebada"] if ("cebada" in data1) else 0
chocolate = data1["chocolate"] if ("chocolate" in data1) else 0
etiqueta = data1["etiqueta"] if ("etiqueta" in data1) else 0
lata_cerveza = data1["lata_cerveza"] if ("lata_cerveza" in data1) else 0
limon = data1["limon"] if ("limon" in data1) else 0
```

```
lupulo = data1["lupulo"] if ("lupulo" in data1) else 0
mango = data1["mango"] if ("mango" in data1) else 0
papaya = data1["papaya"] if ("papaya" in data1) else 0
piña = data1["piña"] if ("piña" in data1) else 0
trigo = data1["trigo"] if ("trigo" in data1) else 0

# La clave primaria no se indica en el insert ya que es el timestamp actual.
cursor.execute(
    "INSERT INTO " +
        "`Inputs`(" +
        "`botellas`,`cebada`,`chocolate`,`etiqueta`," +
        "`lata_cerveza`,`limon`,`lupulo`,`mango`,`papaya`," +
        "`piña`,`trigo`" +
    ") " +
    "VALUES({},{},{},{},{},{},{},{},{},{},{})".format(
        botellas, cebada, chocolate, etiqueta,
        lata_cerveza, limon, lupulo, mango,
        papaya, piña, trigo
    )
)

# Confirmamos la inserción.
db.commit()

# Seleccionamos todos los registros de la tabla.
cursor.execute("SELECT * FROM `Inputs`")

# Creamos un objeto iterador para que se mueva sobre el resultado.
rows = cursor.fetchall();

print(
    "Time - Botellas - Cebada - Chocolate - Etiqueta " +
    "Lata Cerveza - Limon - Lupulo - Mango - Papaya - Piña - Trigo\n"
)

# Mostramos todos los registros.
for row in rows:
    ␣
 ↪print("************************************************************************")
    print(row[0], row[slice(1,len(row))])
```

```
Time - Botellas - Cebada - Chocolate - Etiqueta Lata Cerveza - Limon - Lupulo -
Mango - Papaya - Piña - Trigo

************************************************************************
2020-05-19 23:51:38 (0, 5000, 50, 0, 500, 800, 0, 0, 0, 0, 9000)
```

4. Consulta de las salidas de productos (Output Transfers).

[22]:
```
colulantCervecero = data2["coulant cervecero"] if ("coulant cervecero" in␣
 ↪data2) else 0
exotic = data2["exotic"] if ("exotic" in data2) else 0
lemonSpirit = data2["lemon Spirit"] if ("lemon Spirit" in data2) else 0
tropical = data2["tropical"] if ("tropical" in data2) else 0

# La clave primaria no se indica en el insert ya que es el timestamp actual.
cursor.execute(
    "INSERT INTO " +
        "`Outputs`(" +
        "`coulant cervecero`,`exotic`,`lemon Spirit`,`tropical`"
    ") " +
    "VALUES({},{},{},{})".format(colulantCervecero,exotic,lemonSpirit,tropical)
)

# Confirmamos la inserción.
db.commit()

# Seleccionamos todos los registros de la tabla.
cursor.execute("SELECT * FROM `Outputs`")

# Creamos un objeto iterador para que se mueva sobre el resultado.
rows = cursor.fetchall();

print("Time - Botellas - Cebada - Chocolate - Etiqueta\n")

# Mostramos todos los registros.
for row in rows:
   ␣
 ↪print("**************************************************************************")
    print(row[0], row[slice(1,len(row))])
```

```
Time - Botellas - Cebada - Chocolate - Etiqueta

**************************************************************************
2020-05-19 23:51:38 (600, 200, 600, 600)
```

5. Cierre de la conexión.

[23]:
```
# Cerramos el cursor y la conexión a la BDD.
cursor.close()
db.close()
```