Masmedicos Extender Funcionalidades

automatizar la lectura de archivos de respuesta para cobros realizados, generar el certificado de la póliza de Bancolombia, validar el reporte SARLAFT y notificar al coordinador de call center en caso de que el reporte sea negativo. Además, en caso de que los cobros sean rechazados, se genera un archivo plano con la información correspondiente para enviar a Bancolombiay luego demanera manual actualizar las cuentas de los clientes.

Autor

- PETI Soluciones Productivas

Dependencias

Se requieren los modulos

```
web_sale_extended
product
sale
account
```

Documentación

models

En la carpeta models se encontraran los modelos del modulo.

Clase `AccountMove`

La clase `AccountMove` hereda del modelo `Model` y agrega un campo para Número de intentos de cobro automático.

```
class AccountMove(models.Model):
    _inherit = 'account.move'

    nro_intentos = fields.Integer(string='Intentos de cobro automatico', default=1, copy=False)
```

Este metodo notifica al centro de contacto en caso de un rechazo de Banco de Colombia.

```
    def notify_contact_center_rechazo_bancolombia(self, codigo):
        if self.env.company.contact_center_id:
            ctx = {
                'rechazo': respuestas[codigo],
            }
            template = self.env.ref('web_sale_masmedicos.email_template_rechazo_bancolombia_liquidaciones')
            self.env['mail.template'].browse(template.id).with_context(ctx).send_mail(self.id)
```

Clase `ProductCategory`

La clase `ProductCategory` hereda del modelo de `categorias de producto` y agrega los campos

- `certified_policy_id` campo para almacenar el ID de la vista de certificado de póliza asociada a la categoría de producto.

- `policy_type` Campo para almacenar el tipo de póliza asociada a la categoría de producto. Las opciones son
  'individual' para pólizas individuales y 'collective' para pólizas colectivas.

```
class ProductCategory(models.Model):
    _inherit = 'product.category'

    certified_policy_id = fields.Many2one('ir.ui.view', string='Certificado de poliza')
    policy_type = fields.Selection([('individual', 'Individual'), ('collective', 'Colectiva')],
                                   string='Tipo de poliza')
```

Clase `ResCompany`

La clase `ResCompany` hereda del modelo de la `compañia` y agrega los campos

- `contact_center_id` Campo que almacena el ID del partner que representa al Coordinador del Contact Center asociado a la empresa.
- `sftp_server_env_bancolombia` Campo que permite seleccionar el entorno del servidor SFTP de Bancolombia. Las opciones son 'test' para entorno de prueba y 'prod' para entorno de producción.
- `sftp_hostname_bancolombia` Campo que almacena el nombre de host del servidor SFTP de Bancolombia al que se conectará para descargar los archivos correspondientes.
- `sftp_port_bancolombia` Campo que almacena el número de puerto que se utilizará para conectarse al servidor SFTP de Bancolombia.
- `sftp_user_bancolombia` Campo que almacena el nombre de usuario que se utilizará para autenticarse en el servidor SFTP de Bancolombia.
- `sftp_password_bancolombia` Campo que almacena la contraseña que se utilizará para autenticarse en el servidor SFTP de Bancolombia.
- `sftp_path_output_bancolombia` Campo que almacena el directorio en el servidor SFTP de Bancolombia donde se guardarán los archivos de salida generados por el proceso.
- `sftp_hostname_QA_bancolombia` Campo que almacena el nombre de host del servidor SFTP de prueba de Bancolombia al que se conectará para descargar los archivos correspondientes.
- `sftp_port_QA_bancolombia` Campo que almacena el número de puerto que se utilizará para conectarse al servidor SFTP de prueba de Bancolombia.
- `sftp_user_QA_bancolombia` Campo que almacena el nombre de usuario que se utilizará para autenticarse en el servidor SFTP de prueba de Bancolombia.
- `sftp_password_QA_bancolombia` Campo que almacena la contraseña que se utilizará para autenticarse en el servidor SFTP de prueba de Bancolombia.
- `sftp_path_output_QA_bancolombia` Campo que almacena el directorio en el servidor SFTP de prueba de Bancolombia donde se guardarán los archivos de salida generados por el proceso.

```
class SaleOrder(models.Model):
    _inherit = 'res.company'

    contact_center_id = fields.Many2one('res.partner', string='Coordinador Contact Center')
```

```
    sftp_server_env_bancolombia = fields.Selection(
        [("test", "QA - Test"), ("prod", "Producción")]
    )
    sftp_hostname_bancolombia = fields.Char(string="Hostname SFTP", groups='base.group_user')
    sftp_port_bancolombia = fields.Char(string="Puerto Servidor", groups='base.group_user')
    sftp_user_bancolombia = fields.Char(string="Usuario", groups='base.group_user')
    sftp_password_bancolombia = fields.Char(string="Contraseña", groups='base.group_user')
    sftp_path_output_bancolombia = fields.Char(string="Directorio archivo de salida", groups='base.group_user')

    sftp_hostname_QA_bancolombia = fields.Char(string="Hostname SFTP QA", groups='base.group_user')
    sftp_port_QA_bancolombia = fields.Char(string="Puerto Servidor QA", groups='base.group_user')
    sftp_user_QA_bancolombia = fields.Char(string="Usuario QA", groups='base.group_user')
    sftp_password_QA_bancolombia = fields.Char(string="Contraseña QA", groups='base.group_user')
    sftp_path_output_QA_bancolombia = fields.Char(string="Directorio archivo de salida QA",
groups='base.group_user')
```

Clase `SaleOrder`

La clase SaleOrder es una extensión del modelo existente sale.order de Odoo. La clase SaleOrder agrega dos nuevos campos al modelo base: `tusdatos_reported` y `nro_intentos`.

```
class SaleOrder(models.Model):
    _inherit = 'sale.order'

    tusdatos_reported = fields.Boolean('Reportado', default=False, help='Este campo será verdadero cuando se
encuentre en la lista ONU o OFAC')
    nro_intentos = fields.Integer(string='Intentos de cobro automatico', default=1, copy=False)
```

- El campo `tusdatos_reported` es un campo booleano que se utiliza para indicar si la orden de venta se encuentra en la lista de restricciones de la Oficina de Control de Activos Extranjeros (OFAC) o de la Organización de las Naciones Unidas (ONU).
- El campo `nro_intentos` es un campo entero que se utiliza para rastrear la cantidad de intentos de cobro automático que se han realizado para esta orden de venta.

metodo `conection_sftp`

La función `conection_sftp` se utiliza para establecer una conexión SFTP con un servidor remoto en función de la configuración de la compañía en la que se está trabajando. La función devuelve un objeto de conexión SFTP y la ruta de salida correspondiente.

```
    def conection_sftp(self):
        company = self.env.company
        if company.sftp_server_env_bancolombia == 'test':
            host = company.sftp_hostname_QA_bancolombia
            port = int(company.sftp_port_QA_bancolombia)
            password = company.sftp_password_QA_bancolombia
            user = company.sftp_user_QA_bancolombia
            path = company.sftp_path_output_QA_bancolombia
        elif company.sftp_server_env == 'prod':
            host = company.sftp_hostname_bancolombia
            port = int(company.sftp_port_bancolombia)
            password = company.sftp_password_bancolombia
            user = company.sftp_user_bancolombia
            path = company.sftp_path_output_bancolombia
        else:
            raise UserError(_('No hay un servidor SFTP configurado'))
```

```
    return sftp_utils.connect(host, user, password, port), path
```

metodo `_cron_read_files_sftp`

Este metodo se ejecuta a traves de una accion planificada llamada `Archivos salida SFTP Bancolombia` la cual se ejecuta diariamente

```
def _cron_read_files_sftp(self):
    current_date = fields.Datetime.now()
    connection, path = self.conection_sftp()
```

Se utiliza la conexión establecida anteriormente para abrir el directorio en el servidor y listar los archivos presentes en ese directorio. A continuación, se recorre la lista de archivos y se verifica si el archivo no comienza con 'OK_'. Si es así, se abre el archivo y se leen todas las líneas. Se inicializa la lista lines_cuentas_incorrectas como una lista vacía y se agrega la primera línea del archivo a esa lista si el archivo tiene más de una línea.

```
    with connection as sftp:
        filelist = sftp.listdir(path)
        lines_cuentas_incorrectas = []
        for filename in filelist:
            if not filename.startswith('OK_'):
                with sftp.open(path + filename) as f:
                    lines = f.readlines()
                    lines_cuentas_incorrectas = [lines[0]] if len(lines) > 1 else []
```

Inicia un bucle sobre las líneas restantes de la lista `lines`, se extrae el código de referencia y respuesta de la línea actual en el bucle. Se busca una orden de venta en el entorno actual cuyo nombre sea igual a `referencia` y lo asigna a la variable `order`.

```
for line in lines[1:]:
    referencia = line[80:110].strip()
    codigo_respuesta = line[171:174].strip()
    beneficiary_list = []
    order = self.env['sale.order'].search([('name', '=ilike', referencia)])
```

Se comprueba si la orden existe y se incrementa el campo nro_intentos de la orden en 1. Luego se comprueba que la respuesta se encuentre dentro de la lista la cual indica si fue una respuesta positiva por parte del banco

```
if order:
    order.write({'nro_intentos': order.nro_intentos + 1})
    if codigo_respuesta in ['OK0', 'OK1', 'OK2', 'OK3', 'OK4']:
```

Si el valor de `order.tusdatos_send` es falso, llama al método `get_status_tusdatos_order()` en el objeto order.

```
if not order.tusdatos_send:
    order.get_status_tusdatos_order()
```

Si el valor de `order.tusdatos_approved` es verdadero, llama al método `action_confirm()` en el objeto order, lo que confirma el pedido y genera la suscripción, y llama al método `_send_order_confirmation_mail()`, lo que envía un correo electrónico de confirmación del pedido.

```
if order.tusdatos_approved:
    order.action_confirm()
```

```
    order._send_order_confirmation_mail()
```

Llama al método `write()` en los objetos `partner_id`, `beneficiary0_id`, `beneficiary1_id`, `beneficiary2_id`, `beneficiary3_id`, `beneficiary4_id`, `beneficiary5_id` y `beneficiary6_id`, lo que escribe el ID de suscripción en cada uno de ellos.

```
order.partner_id.write({
    'subscription_id': order.subscription_id.id
})

order.beneficiary0_id.write({
    'subscription_id': order.subscription_id.id
})

order.beneficiary1_id.write({
    'subscription_id': order.subscription_id.id
})

order.beneficiary2_id.write({
    'subscription_id': order.subscription_id.id
})

order.beneficiary3_id.write({
    'subscription_id': order.subscription_id.id
})

order.beneficiary4_id.write({
    'subscription_id': order.subscription_id.id
})

order.beneficiary5_id.write({
    'subscription_id': order.subscription_id.id
})

order.beneficiary6_id.write({
    'subscription_id': order.subscription_id.id
})
```

Agrega una tupla a la lista beneficiary_list para cada uno de los objetos partner_id y beneficiaryX_id, y luego llama al método write() en el objeto subscription_id, lo que escribe beneficiary_list en subscription_partner_ids.

```
beneficiary_list.append((4, order.partner_id.id))
beneficiary_list.append((4, order.beneficiary0_id.id))
beneficiary_list.append((4, order.beneficiary1_id.id))
beneficiary_list.append((4, order.beneficiary2_id.id))
beneficiary_list.append((4, order.beneficiary3_id.id))
beneficiary_list.append((4, order.beneficiary4_id.id))
beneficiary_list.append((4, order.beneficiary5_id.id))
beneficiary_list.append((4, order.beneficiary6_id.id))

order.subscription_id.write({
    'subscription_partner_ids': beneficiary_list
})
```

Llama al método _registrar_archivo_pagos() en el objeto order, lo que registra el archivo de pagos.

```
order._registrar_archivo_pagos()
```

Verifica si la respuesta es negativa por parte del banco y notifica al coordinador del Contact Center, si han pasado 3 intentos con respuesta negativa se cancelará la orden y se borrará numero de cuenta

```
elif codigo_respuesta in respuestas:
    if codigo_respuesta in ['D12', 'D03', 'D10', 'R04', 'R17']:
        lines_cuentas_incorrectas.append(line)
    order.notify_contact_center_rechazo_bancolombia(codigo_respuesta)

    body_message = """
                                <b><span style='color:red;'>Respuesta Bancolombia</span></b><br/>
                                <b>Respuesta:</b> %s<br/>
                            """ % (
        json.dumps(respuestas[codigo_respuesta]),
    )
    order.message_post(body=body_message, type="comment")

    if order.nro_intentos > 3:
        order.action_cancel()
        order.write({'buyer_account_type': False, 'buyer_account_number':False})
```

Para el caso que el archivo de pago sea de cobro recuerrente se hace la busqueda en las facturas

```
else:
    move = self.env['account.move'].search([('name', '=ilike', referencia)])
```

Se comprueba si la factura existe y se incrementa el campo nro_intentos de la factura en 1. Luego se comprueba que la respuesta se encuentre dentro de la lista la cual indica si fue una respuesta positiva por parte del banco y se actualiza el campo payulatam_state

```
if move:
    move.write({'nro_intentos': order.nro_intentos + 1})
    if codigo_respuesta in ['OK0', 'OK1', 'OK2', 'OK3', 'OK4']:
        print('recaudo exitoso')
        move.write({'payulatam_state': 'APPROVED'})
```

si la respuesta es negativa por parte del banco y han pasado 3 intentos con respuesta negativa se notifica al coordinador del Contact Center

```
elif codigo_respuesta in respuestas:
    if codigo_respuesta in ['D12', 'D03', 'D10', 'R04', 'R17']:
        lines_cuentas_incorrectas.append(line)

    if move.nro_intentos > 3:
        move.notify_contact_center_rechazo_bancolombia(codigo_respuesta)

        body_message = """<b><span style='color:red;'>Respuesta Bancolombia</span></b><br/>
        <b>Respuesta:</b> %s<br/>""" % (
            json.dumps(respuestas[codigo_respuesta]),
        )
        move.message_post(body=body_message, type="comment")
```

Actualiza el nombre del archivo en el SFTP poniendo un OK_ al inicio para no volver a procesarlo

```
sftp.rename(path + filename, path + "OK_" + filename)
```

Verifica si la lista lines_cuentas_incorrectas tiene mas de un elemento y genera un archivo .txt en una carpeta con la misma estructura del archivo de salida del banco

```
if len(lines_cuentas_incorrectas) > 1:
    try:
        os.stat('tmp/error/')
    except:
        os.makedirs('tmp/error/')
    with open('tmp/error/%s_%s.txt' % ('error', filename), 'w') as file:
        for line in lines_cuentas_incorrectas:
            file.write(line)
```

metodo _registrar_archivo_pagos

La función _registrar_archivo_pagos tiene como objetivo registrar un pago en la tabla payments_report.

metodo get_status_tusdatos_order

El metodo get_status_tusdatos_order obtiene la validación SARLAFT de la orden desde la cual es invocada la función

metodo notify_contact_center

metodo que se usa para notificar al coordinador del contact center cuando la validación del SARLAFT fue negativa

```
def notify_contact_center(self):
    if self.env.company.contact_center_id:
        template = self.env.ref('web_sale_masmedicos.email_template_tus_datos')
        self.env['mail.template'].browse(template.id).send_mail(self.id)
```

metodo notify_contact_center_rechazo_bancolombia

metodo que se usa para notificar al coordinador del contact center cuando el pago fue rechazado por Bancolombia

```
def notify_contact_center_rechazo_bancolombia(self, codigo):
    if self.env.company.contact_center_id:
        ctx = {
            'rechazo': respuestas[codigo],
        }
        template = self.env.ref('web_sale_masmedicos.email_template_rechazo_bancolombia')
        self.env['mail.template'].browse(template.id).with_context(ctx).send_mail(self.id)
```

metodo cron_get_status_tusdatos

El metodo cron_get_status_tusdatos se ejecuta con una accion planificada llamada Validacion Tus datos Mensual y lo que hace es verificar las ordenes que no han enviado peticiones a tusdatos y luego Se tienen en cuenta únicamente ordenes de venta que no esten aprobadas pero que tengan un número de proceso de parte de tusdatos.