Log aceptación políticas de tratamiento de datos y términos y condiciones

Este modulo guarda el registro de los clientes que aceptan las políticas de tratamiento de datos y los terminos y condiciones.
Mostrando los registros en el menu de informes del modulo de ventas en una ventana tipo lista para el control y depuración de los registros recopilados.

Autor

- Peti soluciones productivas

Modelos

Se crea un nuevo modelo para guardar registro de los usuarios que confirman un pedido.
El modelo se llama "log.aceptacion.politicas" y tiene los siguientes campos:

- date = fecha y hora cuando se confirma el pedido
- ip = IP del cliente
- identificacion_cliente = Número de identificación del cliente
- sponsor_id = campo relacionado con el sponsor del producto
- estado = Indica si la orden fue confirmada
- order_id = relaciona el log con la orden de venta
- campo_vacio = Variable que indica si un usuario puede acceder a los registros de cualquier sponsor
- politicas_name = campo donde se guarda el nombre del archivo de las políticas de tratamiento de datos
- politicas = se guarda los bytes del archivo
- terminos_condiciones_name = campo donde se guarda el nombre del archivo de terminos y condiciones
- terminos_condiciones = se guarda los bytes del archivo

```
class LogAceptacion(models.Model):
    _name = 'log.aceptacion.politicas'
    _description = "Guarda los datos de los usuarios cuando ingresan a comprar un producto"

    date = fields.Datetime("Fecha y hora")
    ip = fields.Char("IP")
    identificacion_cliente = fields.Char("Identificación Cliente")
    sponsor_id = fields.Many2one("res.partner", "Sponsor")
    estado = fields.Selection(selection=[('no_efectivo', 'No efectivo'), ('efectivo', 'Efectivo')], default='no_efectivo')
    order_id = fields.Many2one("sale.order", "Relación con la venta")
    campo_vacio = fields.Boolean('Campo vacio', default=False)
    politicas_name = fields.Char("Nombre políticas de tratamiento de datos")
    politicas = fields.Binary("Políticas de tratamiento de datos")
    terminos_condiciones_name = fields.Char("Nombre términos y condiciones")
    terminos_condiciones = fields.Binary("Términos y condiciones")
```

Herencias de modelos ya existentes

Se hereda del modelo 'sale.order' para sobreescribir el método action_confirm.
Lo que hace el metodo es buscar el log asociado a la orden de venta y cambiarle el estado del log, pasa de "no efectivo" a "efectivo"

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

    def action_confirm(self):
        log = self.env['log.aceptacion.politicas'].search([("order_id", '=', self.id)])
        log.write({'estado': 'efectivo'})
        super(SaleOrderExtend, self).action_confirm()
```

También se hereda del modelo "product.category" para añadirle los campos de tratamiento de datos y terminos y condiciones.

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

    politicas_name = fields.Char("Nombre políticas de tratamiento de datos")
    politicas = fields.Binary("Políticas de tratamiento de datos")
    terminos_condiciones_name = fields.Char("Nombre términos y condiciones")
    terminos_condiciones = fields.Binary("Términos y condiciones")
```

Vistas, menú y acción planificada

Se crea un menú llamado "Log aceptación poli." en la pestaña de informes del modulo de ventas.
Este menú abrira una vista tipo tree, en donde se verán todos los registros del log.
En esta vista se pueden crear filtro o agrupar por los campos del modelo del log.

También se hereda de la vista "product.product_category_form_view" para añadir los nuevos campos del modelo "product.category".

Por último se crea una acción planificada que se ejecuta una vez cada 30 días. Lo que hace es ejecutar el método "eliminar_no_validos" del modelo "log.aceptacion.politicas" lo que hace es buscar todos los registros que no fueron efectivos con fecha inferior a 30 días antes de la fecha actual en la que se ejecuta.

```
    def eliminar_no_validos(self):
        self.env['log.aceptacion.politicas'].search([('date', '<=', datetime.datetime.now() - timedelta(days=30)),
                                                     ('estado', '=', 'no_efectivo')]).unlink()
```

(Controlador) Método principal que crea el registro del log

Este código hereda de la clase WebsiteSale y sobrescribe el método confirm_order. El método confirm_order se ejecuta cuando un cliente confirma un pedido en una tienda en línea.

El código primero obtiene la dirección IP del cliente y el pedido actual. Luego, se obtendrán algunos valores relacionados con los archivos de términos y condiciones y las políticas de tratamiento de datos. Estos valores incluyen el identificador del patrocinador y el número de documento del cliente.

Finalmente, se crea un registro en un modelo llamado 'log.aceptacion.politicas' y se guardan estos valores. Después de eso, se llama al método confirm_order original de la clase padre y se devuelve su resultado.

```
@http.route(['/shop/confirm_order'], type='http', auth="public", website=True, sitemap=False)
    def confirm_order(self, **post):
        ip_address = request.httprequest.environ['REMOTE_ADDR']
        order = request.website.sale_get_order()
        terminos_condiciones = order.order_line[0].product_id.categ_id.terminos_condiciones
        terminos_condiciones_name = order.order_line[0].product_id.categ_id.terminos_condiciones_name
        politicas = order.order_line[0].product_id.categ_id.politicas
        politicas_name = order.order_line[0].product_id.categ_id.politicas_name
        vals = {'date': datetime.datetime.now(), 'ip': ip_address, 'order_id': order.id,
                'sponsor_id': order.order_line[0].product_id.categ_id.sponsor_id.id,
                'identificacion_cliente': order.partner_id.identification_document,
                'terminos_condiciones': terminos_condiciones, 'politicas': politicas,
                'terminos_condiciones_name': terminos_condiciones_name, 'politicas_name': politicas_name}
        request.env['log.aceptacion.politicas'].create(vals)
        res = super(WebsiteSaleExtended, self).confirm_order(**post)
        return res
```