LISTING PROGRAM

dms_stockcard.py: Program back end dari sistem data mart

```
# 1: imports of python lib
# 2: import of known third party lib
# 3:  imports of odoo
from odoo import models, fields, api
from odoo.exceptions import Warning
# 4:  imports from odoo modules
import json
import requests
import logging
_logger = logging.getLogger(__name__)
# 5: local imports
# 6: Import of unknown third party lib
class StockCard(models.Model):
 _name = "dms.stockcard"
_description = 'Stock Card Sparepart'
_order = 'write_date desc'
# 7: defaults methods
@api.model
def _get_default_date(self):
return self.env['res.branch'].get_default_date()
def _get_default_datetime(self):
return self.env['res.branch'].get_default_datetime()
# 8: fields
name = fields.Char(string="Name", readonly=True)
date = fields.Date(string="Date", default=_get_default_date, readonly=True)
state = fields.Selection([
 ('draft', 'Draft'),
 ('ready', 'On Progres'),
 ('assigned', 'Ready to Transfer'),
 ('transfered', 'Transfered')], string="Status", readonly=True, default='draft') transaction_type =
fields.Selection(['incoming','Incoming'), ('outgoing','Outgoing')], string="Transaction Type",
readonly=True)
# Audit Trail
confirm_uid = fields.Many2one('res.users','Confirmed by')
confirm_date = fields.Datetime('Confirmed on')
transfer_uid = fields.Many2one('res.users','Transfer by')
transfer_date = fields.Datetime('Transfer on')
cancel_uid = fields.Many2one('res.users','Canceled by')
cancel_date = fields.Datetime('Canceled on')
 # 8: relation fields
picking_ids = fields.Many2many('stock.picking', string="Source Document", required=True,
readonly=True, states={'draft': [('readonly', False)]}, domain=[])
line_ids = fields.One2many('dms.stockcard.line.transaction', 'stockcard_id', string="List
Sparepart", readonly=True, states={'draft': [('readonly', False)]})

branch_id = fields.Many2one('res.branch', string="Branch", required=True, readonly=True,
states={'draft': [('readonly', False)]}, domain=[])
# 9: constraints & sql constraints
# 10: compute/depends & on change methods

@api.onchange('picking_ids')
def _onchange_picking_ids(self):
self.line_ids = False
lines = []
```

```
for line in self.picking_ids.move_lines:
 if line.qty_available != 0:
 binbox_line_obj
=self.env['dms.stockcard.line.binbox'].suspend_security().search([('product_id','=',line.product_id.i
d)],limit=1,order='write_date desc')
 indexing_part_obj
=self.env['dms.stockcard.indexing.part.line'].suspend_security().search([('kode_dus','=',line.origin)
],limit=1)
 self.transaction_type == 'incoming':
 vals = {'picking_id':line.picking_id.id, 'product_id':line.product_id, 'qty':line.qty_available,
'qty_available':line.product_uom_qty, 'kode_dus':line.origin,
'binbox_id':binbox_line_obj.binbox_id, 'lokasi_sementara':indexing_part_obj.index_id.name,
'no_index':indexing_part_obj.no_index,
 lines.append((0,0,vals))
if self.transaction_type == 'outgoing':
 vals = {'picking_id':line.picking_id.id, 'product_id':line.product_id, 'qty':line.qty_available,
'qty_available':line.product_uom_qty, 'kode_dus':line.origin,
'binbox_id':binbox_line_obj.binbox_id }
lines.append((0,0,vals))
 self.line_ids = lines

@api.onchange('transaction_type')
 def _onchange_transaction(self):
 if self.transaction_type == 'incoming':
 self.branch_id = self.env['res.branch'].suspend_security().search([('kode_dealer', '=', 'MML')],
limit=1).id
self.picking_ids = False
domain ={'picking_ids':[('transaction_type','=','incoming')],'branch_id':[('kode_dealer','=','MML')]}
return {'domain':domain}

if self.transaction_type == 'outgoing':
 self.branch_id = False
self.picking_ids = False
domain =
{'picking_ids':[('transaction_type','=','outgoing')],'branch_id':[('kode_dealer','!=','MML')]}
 return {'domain':domain}

@api.onchange('branch_id')
 def _onchange_branch(self):
 self.picking_ids = False
if self.branch_id:
domain = {'picking_ids':[('branch_id','=',self.branch_id.id),('state','=','assigned')]} return
{'domain':domain}
# 11: override methods

@api.model
def create(self, vals):
 branch_src = self.env['res.branch'].suspend_security().search([('id','=',vals['branch_id'])],limit=1)
doc_code = branch_src.kode_dealer
vals['name'] = self.env['ir.sequence'].suspend_security().get_per_doc_code(doc_code,'SC')
return super(StockCard, self).create(vals)

 def unlink(self):
 for x in self:
 if x.state != 'draft':
 raise Warning('Perhatian!\nData tidak bisa dihapus.')
 indexing_part_obj =
```

```
self.env['dms.stockcard.indexing.part'].suspend_security().search([('stockcard_id','=',self.id)])
if self.transaction_type == 'incoming':
   if indexing_part_obj:
      raise Warning('Perhatian!\nData tidak bisa dihapus.')
 return super(StockCard, self).unlink() \
 # 12: action methods
def action_confirm(self):
 for record in self.line_ids:
 # Olah data QTY Stock Move
qty_reserved = record.stock_move_id.qty_reserved + record.qty
qty_available = record.stock_move_id.qty_available - record.qty
record.stock_move_id.write({'qty_reserved': qty_reserved, 'qty_available': qty_available})
if record.stock_move_id.qty_reserved > record.stock_move_id.product_uom_qty:
raise Warning ("Qty Reserved pada produk '%s' melebihi Qty Seharusnya,\nPeriksa kembali
Transaksi anda!"%(record.product_id.default_code))
 self.write({'state': 'ready', 'confirm_uid':self.env.user.id,
'confirm_date':self._get_default_datetime()})
def action_cancel(self):
 for record in self.line_ids:
 # Olah data QTY Stock Move
qty_reserved = record.stock_move_id.qty_reserved - record.qty
qty_available = record.stock_move_id.qty_available + record.qty
record.stock_move_id.write({'qty_reserved': qty_reserved, 'qty_available': qty_available})
self.write({'state': 'draft', 'cancel_uid':self.env.user.id, 'cancel_date':self._get_default_datetime()})

def action_transfer(self):
 for record in self.line_ids:
 if self.transaction_type == 'outgoing':
 if record.state != 'done':
 raise Warning ("Sparepart %s belum melewati proses QC!" %(record.product_id.default_code))
 if self.transaction_type == 'incoming':
stockcard_line =
self.env['dms.stockcard.line.transaction'].suspend_security().search([('stockcard_id','=',self.id),
('binbox_id','=',False)])
 if stockcard_line:
 for rec in stockcard_line:
raise Warning ("Lokasi pada Sparepart '%s' tidak boleh kosong, Periksa kembali transaksi
anda!"%(rec.product_id.default_code))

 # Olah data Stock Move
qty_reserved = record.stock_move_id.qty_reserved - record.qty
qty_done = record.stock_move_id.qty_done + record.qty
state = 'assigned'
if record.stock_move_id.qty_done + record.qty == record.stock_move_id.product_uom_qty:
 state = 'done'
record.stock_move_id.write({'qty_done': qty_done, 'qty_reserved': qty_reserved, 'state':state})

# Cek data line binbox
binbox_line_obj =
self.env['dms.stockcard.line.binbox'].suspend_security().search([('product_id','=',record.product_id
.id)
'binbox_id','=',record.binbox_id.id)],limit=1)
```