Subject: Verschenke 80.000€
Posted by daniel91 on Wed, 06 Jun 2018 19:48:12 GMT
View Forum Message <> Reply to Message

Bitteschön:

```
import datetime

from flask import jsonify, request, make_response, Blueprint
from flask_babel import gettext as _, format_date
from webapp.models.production import Production
from webapp.models.production_state import ProductionState
from webapp.models.assembly import Assembly
from webapp.models.current_worker import CurrentWorker

from webapp.models.event_log import EventLog
from webapp.app import db
from webapp import exceptions
from webapp.helper import ensure_in_request_data, remove_leading_zeroes
from webapp.logger import logger
from webapp.helper import errorhandler, format_time_ticket
from webapp.SAPConnector.sap_conector import SAPConnector

blueprint_api_sap = Blueprint('api_sap', __name__)
blueprint_api_sap.errorhandler(exceptions.DA645Exception)(errorhandler)


@blueprint_api_sap.route('/production/orders/released', methods=['GET'])
def api_sap_get_released_orders():
    sap_api_connector = SAPConnector()
    released_orders = sap_api_connector.get_released_orders()
    return make_response(jsonify(released_orders), 200)


@blueprint_api_sap.route('/production/orders/released/table', methods=['GET'])
def api_sap_get_released_orders_table():
    sap_api_connector = SAPConnector()
    released_orders = sap_api_connector.get_released_orders()
    return make_response(jsonify(released_orders), 200)


@blueprint_api_sap.route('/production/orders/released/<string:capacity_number>',
methods=['GET'])
def api_sap_get_released_orders_by_capacity_id(capacity_number):
    sap_api_connector = SAPConnector()
    sap_released_orders = sap_api_connector.get_released_orders()
    released_orders = []
```

```
    for sap_released_order in sap_released_orders['IT_RELEASED_ORDERS_JSON']:
        if sap_released_order['ARBPL'] == capacity_number:
            released_orders.append(sap_released_order)

    return make_response(jsonify(released_orders), 200)


@blueprint_api_sap.route('/production/orders/released/<string:capacity_number>/table',
methods=['GET'])
def api_sap_get_released_orders_table_by_capacity_id(capacity_number):
    sap_api_connector = SAPConnector()
    sap_released_orders = sap_api_connector.get_released_orders()
    released_orders = {'data': []}

    for sap_released_order in sap_released_orders['IT_RELEASED_ORDERS_JSON']:
        if sap_released_order['ARBPL'] != capacity_number:
            continue

        sap_order_details = sap_api_connector.get_order_details(sap_released_order['AUFNR'])
        sap_operation = None
        operation_sequence = 0

        for operation_sequence in range(operation_sequence,
len(sap_order_details['LT_OPERATION'])):
            if sap_order_details['LT_OPERATION'][operation_sequence]['OPERATION_NUMBER'] ==
sap_released_order['VORNR']:
                sap_operation = sap_order_details['LT_OPERATION'][operation_sequence]
                break

        if sap_operation is None:
            raise exceptions.SAPNotFound(_('Operation not found'))

        start_date = datetime.datetime.strptime(sap_released_order['SSAVD'], '%Y-%m-%d')
        end_date = datetime.datetime.strptime(sap_released_order['SSEDD'], '%Y-%m-%d')

        row = [remove_leading_zeroes(sap_released_order['AUFNR']),
               sap_released_order['VORNR'],
               remove_leading_zeroes(sap_operation['CONF_NO']),
               sap_operation['DESCRIPTION'] + " " + sap_operation['DESCRIPTION2'],
               sap_released_order['LTXA1'],
               format_date(start_date, format='short'),
               format_date(end_date, format='short'),
               sap_released_order['STTXT'],
               ]

        if operation_sequence == 0:
            row.append('table-success')
        else:
```

```
            sap_previous_operation = sap_order_details['LT_OPERATION'][operation_sequence - 1]
            sap_previous_operation_status_list = sap_previous_operation['SYSTEM_STATUS'].split('
')
            if 'CNF' in sap_previous_operation_status_list:
                row.append('table-success')
            elif 'PCNF' in sap_previous_operation_status_list and
sap_previous_operation['QUANTITY'] > 0:
                row.append('table-warning')
            else:
                row.append('table-danger')

        released_orders['data'].append(row)

    return make_response(jsonify(released_orders), 200)


@blueprint_api_sap.route('/hr/authenticate/<string:worker_id>', methods=['GET'])
def api_sap_authenticate_worker(worker_id):
    worker_id_expected_length = 8

    if not worker_id.isdigit() or len(worker_id) > worker_id_expected_length:
        raise exceptions.RequestDataInvalidWorkerId(_('Invalid worker id'))

    if len(worker_id) < worker_id_expected_length:
        worker_id = "0" * (worker_id_expected_length - len(worker_id)) + worker_id

    sap_api_connector = SAPConnector()
    sap_worker = sap_api_connector.get_worker_by_id(worker_id)

    worker = {
        "id": worker_id,
        "name": "{forename} {lastname}".format(
            forename=sap_worker["WA_EMPLOYEE_JSON"]["VORNA"],
            lastname=sap_worker["WA_EMPLOYEE_JSON"]["NACHN"]
        )
    }
    return make_response(jsonify(worker), 200)


@blueprint_api_sap.route('/production/operation/start',
                         methods=['POST'])
def api_sap_start_operation():
    request_data = request.get_json()

    ensure_in_request_data(request_data, ("production_id", int, lambda x: x > 0),
                           error=exceptions.RequestDataInvalidProductionId(_('Invalid production id')))

    ensure_in_request_data(request_data, ('operation_confirmation_number', str, lambda x: len(x)
```

```
> 0),
                    error=exceptions.RequestDataInvalidOperationNumber(_('Invalid operation
number')))

    ensure_in_request_data(request_data, ("worker.id", str, lambda x: len(x) is 8),
                    error=exceptions.RequestDataInvalidWorkerId(_('Invalid worker id')))

    production = Production.query.get_or_404(request_data['production_id'])
    production_state = ProductionState.query.filter_by(name='Production').first_or_404()

    if production.assembly:
        raise exceptions.RequestInvalidassemblyState(_('Assembly already started'))

    request_worker = request_data["worker"]
    request_operation_confirmation_number = request_data["operation_confirmation_number"]
    operation_confirmation_number_expected_length = 10

    current_worker_production =
CurrentWorker.query.filter_by(worker_id=request_worker["id"]).first()

    if current_worker_production:
        raise exceptions.RequestInvalidWorkerState(_('{worker_name} already working on
{production_name}'.format(
            worker_name=current_worker_production.worker_name,
            production_name=current_worker_production.assembly.production.name)))

    if len(request_operation_confirmation_number) <
operation_confirmation_number_expected_length:
        request_operation_confirmation_number = "0" *
(operation_confirmation_number_expected_length - len(
            request_operation_confirmation_number)) + request_operation_confirmation_number

    sap_api_connector = SAPConnector()
    sap_timeticket_proposal =
sap_api_connector.get_timeticket_proposal(request_operation_confirmation_number)
    sap_order_details =
sap_api_connector.get_order_details(sap_timeticket_proposal['LT_TIMETICKETS'][0]['ORDERID'
])

    sap_order_details_operation = None

    for sap_operation in sap_order_details['LT_OPERATION']:
        if sap_operation['CONF_NO'] == request_operation_confirmation_number:
            sap_order_details_operation = sap_operation
            break

    if not sap_order_details_operation:
        raise exceptions.SAPNotFound(_('Confirmation number not in SAP order details'))
```

```
    if production.capacity_number != sap_order_details_operation['WORK_CENTER']:
        raise exceptions.RequestDataInvalidOperationState(_('Operation not assigned to this
capacity'))

    production_already_assembling = Production.query.join(Assembly).filter(
        Assembly.order_number == sap_order_details['LS_HEADER']['ORDER_NUMBER']).filter(
        Assembly.operation_confirmation_number != request_operation_confirmation_number).first()

    if production_already_assembling:
        raise exceptions.RequestDataInvalidOperationState(_(
            'Another Operation of order {order_number} already in assembly at
{production_name}'.format(

order_number=remove_leading_zeroes(production_already_assembling.assembly.order_number)
,
                production_name=production_already_assembling.name
            )))

    sap_order_details_operation_status = sap_order_details_operation['SYSTEM_STATUS'].split('
')

    if 'REL' not in sap_order_details_operation_status:
        raise exceptions.RequestDataInvalidOperationState(_('Operation not released'))

    if 'CNF' in sap_order_details_operation_status:
        raise exceptions.RequestDataInvalidOperationState(_('Operation already confirmed'))

    assembly = Assembly()
    assembly.order_number = sap_order_details['LS_HEADER']['ORDER_NUMBER']
    assembly.operation_confirmation_number = sap_order_details_operation['CONF_NO']
    assembly.production_id = Production.id

    production.production_state = production_state
    production.assembly = assembly

    try:
        db.session.add(assembly)
        db.session.flush()
        current_worker = CurrentWorker(worker_name=request_worker["name"],
                                       worker_id=request_worker["id"],
                                       assembly_id=assembly.id)
        db.session.add(current_worker)
        db.session.flush()

        ev_start_op = EventLog(event='start_operation', value=
        {"assembly_id": assembly.id,
         "order_number": assembly.order_number,
```

```
        "operation_confirmation_number": assembly.operation_confirmation_number,
        "production_id": production.id,
        "production_name": production.name,
        "production_workplace_number": production.workplace_number,
        "worker_id": current_worker.worker_id})

      ev_prod_state = EventLog(event='production_state', value=
      {"production_state_id": production.production_state_id,
        "production_state_name": production.production_state.name,
        "production_id": production.id,
        "production_name": production.name,
        "production_workplace_number": production.workplace_number,
        "worker_id": current_worker.worker_id})

      ev_start_work = EventLog(event="start_work", value=
      {"assembly_id": production.assembly.id,
        "order_number": assembly.order_number,
        "operation_confirmation_number": assembly.operation_confirmation_number,
        "production_id": production.id,
        "production_name": production.name,
        "production_workplace_number": production.workplace_number,
        "worker_id": current_worker.worker_id})

      db.session.add(ev_start_op)
      db.session.add(ev_prod_state)
      db.session.add(ev_start_work)
      db.session.add(current_worker)

      db.session.commit()

    except Exception as e:
      logger.error(str(e))
      raise exceptions.RequestGenericError(str(e))

    return make_response(jsonify(production.get_dict()), 200)


@blueprint_api_sap.route('/production/operation/finish',
                         methods=['POST'])
def api_sap_finish_operation():
    request_data = request.get_json()

    ensure_in_request_data(request_data, ("production_id", int, lambda x: x > 0),
                           error=exceptions.RequestDataInvalidProductionId(_('Invalid production id')))

    ensure_in_request_data(request_data, ("worker.id", str, lambda x: len(x) is 8),
                           error=exceptions.RequestDataInvalidWorkerId(_('Invalid worker id')))
    ensure_in_request_data(request_data, ("parts", int, lambda x: x >= 0),
```

```
                    error=exceptions.RequestDataInvalidPartAmount(_('Invalid part amount')))

    production = Production.query.get_or_404(request_data['production_id'])

    request_worker = request_data['worker']

    if not production.assembly:
        raise exceptions.RequestInvalidassemblyState(_('No assembly in progress'))

    current_worker = CurrentWorker.query.join(Assembly).filter(
        CurrentWorker.assembly_id == production.assembly.id,
        CurrentWorker.worker_id == request_worker[
            "id"]).first()

    if not current_worker:
        raise exceptions.RequestInvalidWorkerState(
            _('{worker_name} not working on
{production_name}'.format(worker_name=request_worker['name'],
                                                    production_name=production.name)))

    request_parts = request_data['parts']

    sap_api_connector = SAPConnector()
    sap_timeticket_proposal = sap_api_connector.get_timeticket_proposal(
        production.assembly.operation_confirmation_number)

    if request_parts > sap_timeticket_proposal['LT_TIMETICKETS'][0]['YIELD']:
        raise exceptions.SAPException(_('Maximum bookable parts {parts_yield} exceeded by
{diff_parts}'.format(
            parts_yield=int(sap_timeticket_proposal['LT_TIMETICKETS'][0]['YIELD']),
            diff_parts=int(request_parts - sap_timeticket_proposal['LT_TIMETICKETS'][0]['YIELD'])
        )))

    assemblies_to_finish = [production.assembly]

    if request_parts == sap_timeticket_proposal['LT_TIMETICKETS'][0]['YIELD']:
        assemblies_to_finish = Assembly.query.join(Production).filter(
            Assembly.operation_confirmation_number ==
production.assembly.operation_confirmation_number).all()

    time_tickets_jar = []

    for assembly_to_finish in assemblies_to_finish:
        for assembly_to_finish_worker in assembly_to_finish.current_worker:
            if assembly_to_finish_worker.id != current_worker.id or production.id !=
assembly_to_finish.production.id:
                time_ticket_assembly_worker = format_time_ticket(sap_timeticket_proposal,
                                                 assembly_to_finish_worker.worker_id,
```

```
                                        's',
                                        assembly_to_finish_worker.get_work_time_seconds(),
                                        0,
                                        assembly_to_finish.production.printer_id
                                        )
                time_tickets_jar.append(time_ticket_assembly_worker)

    time_ticket_current_worker = format_time_ticket(
        sap_timeticket_proposal,
        current_worker.worker_id,
        's',
        current_worker.get_work_time_seconds(),
        request_parts,
        production.printer_id)

    time_tickets_jar.append(time_ticket_current_worker)
    sap_api_connector.create_time_ticket(time_tickets_jar)

    try:
        for assembly_to_finish in assemblies_to_finish:
            for assembly_to_finish_worker in assembly_to_finish.current_worker:
                worker_parts = 0
                if assembly_to_finish.production.id == production.id and current_worker.worker_id ==
request_worker[
                    "id"]:
                    worker_parts = request_parts

                ev_stop_work = EventLog(event="stop_work", value=
                {"assembly_id": assembly_to_finish.id,
                 "order_number": assembly_to_finish.order_number,
                 "operation_confirmation_number": assembly_to_finish.operation_confirmation_number,
                 "parts": worker_parts,
                 "production_id": assembly_to_finish.production.id,
                 "production_name": assembly_to_finish.production.name,
                 "production_workplace_number": assembly_to_finish.production.workplace_number,
                 "worker_id": assembly_to_finish_worker.worker_id})
                db.session.add(ev_stop_work)
                db.session.delete(assembly_to_finish_worker)

            ev_prod_state = EventLog(event='production_state', value=
            {"production_state_id": assembly_to_finish.production.production_state.id,
             "production_state_name": assembly_to_finish.production.production_state.name,
             "production_name": assembly_to_finish.production.name,
             "production_workplace_number": assembly_to_finish.production.workplace_number,
             "worker_id": current_worker.worker_id})

            ev_fin_op = EventLog(event='finish_operation', value=
            {"assembly_id": assembly_to_finish.id,
```

```
            "order_number": assembly_to_finish.order_number,
            "operation_confirmation_number": assembly_to_finish.operation_confirmation_number,
            "parts": request_parts,
            "production_id": assembly_to_finish.production.id,
            "production_name": assembly_to_finish.production.name,
            "production_workplace_number": assembly_to_finish.production.workplace_number,
            "worker_id": current_worker.worker_id})

            db.session.add(ev_prod_state)
            db.session.add(ev_fin_op)
            assembly_to_finish.production.production_state = None
            assembly_to_finish.finish_date = datetime.datetime.utcnow()
            assembly_to_finish.production.assembly = None

        db.session.commit()
    except Exception as e:
        logger.error(str(e))
        raise exceptions.RequestGenericError(str(e))

    return make_response(jsonify(production.get_dict()), 200)


@blueprint_api_sap.route('/production/work/start',
                         methods=['POST'])
def api_sap_start_work():
    request_data = request.get_json()

    ensure_in_request_data(request_data, ("production_id", int, lambda x: x > 0),
                           error=exceptions.RequestDataInvalidProductionId(_('Invalid production id')))

    ensure_in_request_data(request_data, ("worker.id", str, lambda x: len(x) is 8),
                           error=exceptions.RequestDataInvalidWorkerId(_('Invalid worker id')))

    production = Production.query.get_or_404(request_data['production_id'])
    request_worker = request_data["worker"]

    if not production.assembly:
        raise exceptions.RequestInvalidassemblyState(_('No assembly in progress'))

    if CurrentWorker.query.filter(
            CurrentWorker.assembly_id == production.assembly.id,
            CurrentWorker.worker_id == request_worker[
                "id"]).first():
        raise exceptions.RequestInvalidassemblyState(
            _('Worker already working on
{production_name}'.format(production_name=production.name)))

    current_worker_production =
```

```
CurrentWorker.query.filter_by(worker_id=request_worker["id"]).first()

    if current_worker_production:
        raise exceptions.RequestInvalidWorkerState(_('{worker_name} already working on
{production_name}'.format(
            worker_name=current_worker_production.worker_name,
            production_name=current_worker_production.assembly.production.name)))

    current_worker = CurrentWorker(
        worker_name=request_worker["name"],
        worker_id=request_worker["id"],
        assembly_id=production.assembly.id)

    if production.production_state.type == 'error':
        current_worker.interrupt_work_date = datetime.datetime.utcnow()

    try:
        db.session.add(current_worker)
        db.session.flush()

        ev_start_work = EventLog(event="start_work", value={
            "assembly_id": production.assembly.id,
            "order_number": production.assembly.order_number,
            "production_id": production.id,
            "production_name": production.name,
            "production_workplace_number": production.workplace_number,
            "worker_id": current_worker.worker_id})

        db.session.add(ev_start_work)
        db.session.commit()
    except Exception as e:
        logger.error(str(e))
        raise exceptions.RequestGenericError(str(e))

    return make_response(jsonify(production.get_dict()), 200)


@blueprint_api_sap.route('/production/work/stop', methods=['POST'])
def api_sap_stop_work():
    request_data = request.get_json()

    ensure_in_request_data(request_data, ("worker.id", str, lambda x: len(x) is 8),
                           error=exceptions.RequestDataInvalidWorkerId(_('Invalid worker id')))

    ensure_in_request_data(request_data, ("production_id", int, lambda x: x > 0),
                           error=exceptions.RequestDataInvalidProductionId(_('Invalid production id')))

    ensure_in_request_data(request_data, ("parts", int, lambda x: x >= 0),
```

```
                    error=exceptions.RequestDataInvalidPartAmount(_('Invalid part amount')))

    production = Production.query.get_or_404(request_data['production_id'])
    request_worker = request_data["worker"]
    request_parts = request_data["parts"]

    if not production.assembly.id:
        raise exceptions.RequestInvalidassemblyState(_('No assembly in progress'))

    current_worker = CurrentWorker.query.join(Assembly).filter(
        CurrentWorker.assembly_id == production.assembly.id,
        CurrentWorker.worker_id == request_worker[
            "id"]).first()

    if not current_worker:
        raise exceptions.RequestInvalidassemblyState(
            _('{worker_name} not working on
{production_name}'.format(worker_name=request_worker['name'],
                                                    production_name=production.name)))

    sap_api_connector = SAPConnector()
    sap_timeticket_proposal = sap_api_connector.get_timeticket_proposal(
        production.assembly.operation_confirmation_number)

    if request_parts > sap_timeticket_proposal['LT_TIMETICKETS'][0]['YIELD']:
        raise exceptions.SAPException(_('Maximum bookable parts {parts_yield} exceeded by
{diff_parts}'.format(
            parts_yield=int(sap_timeticket_proposal['LT_TIMETICKETS'][0]['YIELD']),
            diff_parts=int(request_parts - sap_timeticket_proposal['LT_TIMETICKETS'][0]['YIELD'])
        )))
    elif request_parts == sap_timeticket_proposal['LT_TIMETICKETS'][0]['YIELD']:
        raise exceptions.SAPException(_('Maximum bookable parts of {parts_yield} would be
reached, '
                                        'please finish Operation'

''.format(parts_yield=int(sap_timeticket_proposal['LT_TIMETICKETS'][0]['YIELD'])
                                            )))

    time_ticket = format_time_ticket(sap_timeticket_proposal,
                                     current_worker.worker_id,
                                     's',
                                     current_worker.get_work_time_seconds(),
                                     request_parts,
                                     production.printer_id)

    sap_api_connector.create_time_ticket([time_ticket])

    try:
```

```
        ev_stop_work = EventLog(event="stop_work", value={
            "assembly_id": production.assembly.id,
            "order_number": production.assembly.order_number,
            "operation_confirmation_number": production.assembly.operation_confirmation_number,
            "parts": request_parts,
            "production_id": production.id,
            "production_name": production.name,
            "production_workplace_number": production.workplace_number,
            "worker_id": current_worker.worker_id})

        db.session.add(ev_stop_work)
        db.session.delete(current_worker)
        db.session.commit()

    except Exception as e:
        logger.error(str(e))
        raise exceptions.RequestGenericError(str(e))

    return make_response(jsonify(production.get_dict()), 200)


@blueprint_api_sap.route('/production/state', methods=['POST'])
def api_set_production_state():
    request_data = request.get_json()

    ensure_in_request_data(request_data, ("production_id", int, lambda x: x > 0),
                           error=exceptions.RequestDataInvalidProductionId(_('Invalid production id')))

    ensure_in_request_data(request_data, ("production_state_id", int, lambda x: x > 0),
                           error=exceptions.RequestDataInvalidProductionStateId(_('Invalid production
state id')))

    ensure_in_request_data(request_data, ("worker.id", str, lambda x: len(x) is 8),
                           error=exceptions.RequestDataInvalidWorkerId(_('Invalid worker id')))

    request_worker = request_data['worker']
    production = Production.query.get_or_404(request_data['production_id'])
    production_state = ProductionState.query.get_or_404(request_data['production_state_id'])
    production.production_state = production_state

    if production.assembly:
        if production_state.type == "ok":
            for current_worker in production.assembly.current_worker:
                current_worker.interrupt_work_date = None
                current_worker.start_work_date = datetime.datetime.utcnow()
        else:
            for current_worker in production.assembly.current_worker:
                if not current_worker.interrupt_work_date:
```

```
                current_worker.interrupt_work_date = datetime.datetime.utcnow()

    try:
        ev_pro_state = EventLog(event='production_state', value=
        {"production_state_id": production.production_state_id,
         "production_state_name": production.production_state.name,
         "production_id": production.id,
         "production_name": production.name,
         "production_workplace_number": production.workplace_number,
         "worker_id": request_worker["id"]})

        db.session.add(ev_pro_state)
        db.session.commit()
    except Exception as e:
        logger.error(str(e))
        raise exceptions.RequestGenericError(str(e))

    return make_response(jsonify(production.get_dict()), 200)
```
