```java
package com.davon.siemens.service;

import java.sql.Timestamp;
import java.util.Calendar;

import org.jetbrains.annotations.NotNull;
import org.jetbrains.annotations.Nullable;
import org.json.JSONArray;
import org.json.JSONObject;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.core.env.Environment;
import org.springframework.stereotype.Service;

import com.davon.dvnfrm.exception.SuUncheckedException;
import com.davon.siemens.exception.MessageProcessCheckedException;

import okhttp3.MediaType;
import okhttp3.OkHttpClient;
import okhttp3.Request;
import okhttp3.RequestBody;
import okhttp3.Response;

@Service
public class UtsQueryService implements UtsQueryServiceInt {

   private static final String SONUC_MESAJI = "sonucMesaji";
   @Autowired
   private Environment env;

   private static final Logger LOGGER = LoggerFactory.getLogger(
           UtsQueryService.class);

   /**
    * Eğer expireDate parameteresi null geliyorsa ve rafOmru bilgisi boş ise.:
    * <br>
    * <br>
    * importDate null değil ise üretim tarihi = importDate - 2 sene<br>
    * importDate null ise üretim tarihi = o an - 2 sene<br>
    * <br>
    * Eğer expireDate parametresi null gelmiyorsa ve rafOmru bilgisi UTS de
var
    * ise : <br>
    *
    * @param siteId
    * @param utsProductNumber
    * @param expireDate
    * @param importDate
```

```
     * @return
     */
    @Override
    public Timestamp generateManufactureDateFromUTS(Integer siteId,
            String utsProductNumber, String lotNumber, Timestamp expireDate,
            Timestamp importDate) throws MessageProcessCheckedException {
        Timestamp result = null;
        String manufactureDate = getManufactureDateOfLotFromUTS(siteId,
                utsProductNumber, lotNumber);
        if (manufactureDate != null) {
            result = Timestamp.valueOf(manufactureDate);
        } else {
            String resStr = searchMaterialUTS(siteId, utsProductNumber);
            if (resStr == null) {
                throw new MessageProcessCheckedException(utsProductNumber
                        + " could not be found in UTS system.");
            } else {
                JSONObject jsonResult = new JSONObject(resStr);
                if ("Ürün bulundu.".equals(jsonResult.get(SONUC_MESAJI))) {
                    JSONArray urunList = jsonResult.getJSONArray(
                            "urunDetayList");
                    final Object rafOmru1 = urunList.getJSONObject(0).get(
                            "rafOmru");
                    if (JSONObject.NULL.equals(rafOmru1)
                            && expireDate == null) {
                        result = calculateManufactureDateUsingImportDate(
                                importDate);
                    } else {
                        final Object rafOmruBirimi = urunList.getJSONObject(0)
                                .get("rafOmruBirimi");
                        result = calculateManufactureDateUsingExpireDate(
                                utsProductNumber, expireDate, rafOmruBirimi,
                                rafOmru1);
                    }
                } else {
                    throw new MessageProcessCheckedException(utsProductNumber
                            + " cannot be found on UTS. UTS msg: " + jsonResult
                                    .get(SONUC_MESAJI));
                }
            }
        }
        return result;

    }

    @NotNull
    private Timestamp calculateManufactureDateUsingImportDate(
            Timestamp importDate) {
        Timestamp resultTimeStamp;
```

```
        // Eğer expireDate parameteresi null GELİYORSA:
        // importDate null değil ise üretim tarihi = importDate
        // - 2 sene
        // importDate null ise üretim tarihi = o an - 2 sene

        Calendar cal = Calendar.getInstance();
        if (importDate != null) {
            cal.setTime(importDate);
        }
        cal.add(Calendar.YEAR, -2);
        resultTimeStamp = new Timestamp(cal.getTime().getTime());
        return resultTimeStamp;
    }

    @Nullable
    private Timestamp calculateManufactureDateUsingExpireDate(
            String utsProductNumber, Timestamp expireDate,
            Object rafOmruBirimi1, Object rafOmru1)
            throws MessageProcessCheckedException {
        String rafOmruBirimi;
        if (expireDate == null) {
            final String msg = utsProductNumber + " expire should not bu null.";
            throw new MessageProcessCheckedException(msg);
        }
        // Eğer expireDate parameteresi null GELMİYORSA:
        if (!JSONObject.NULL.equals(rafOmruBirimi1)) {
            rafOmruBirimi = (String) rafOmruBirimi1;
        } else {
            final String msg = utsProductNumber
                    + " rafOmruBirimi cannot be empty.";
            throw new MessageProcessCheckedException(msg);
        }
        Integer rafOmru;
        if (!JSONObject.NULL.equals(rafOmru1)) {
            rafOmru = (Integer) rafOmru1;
        } else {
            final String msg = utsProductNumber + " rafOmru cannot be empty.";
            throw new MessageProcessCheckedException(msg);
        }

        return calculateResultTimeStamp(utsProductNumber, expireDate,
                rafOmruBirimi, rafOmru);
    }

    @Nullable
    private Timestamp calculateResultTimeStamp(String utsProductNumber,
            Timestamp expireDate, String rafOmruBirimi, Integer rafOmru)
            throws MessageProcessCheckedException {
        Timestamp resultTimeStamp;
```

```
        if (rafOmruBirimi != null && !"".equals(rafOmruBirimi)
                && rafOmru != null) {

            Calendar cal = Calendar.getInstance();
            cal.setTime(expireDate);
            switch (rafOmruBirimi) {
            case "GUN":
                cal.add(Calendar.DATE, -1 * rafOmru);
                break;
            case "AY":
                cal.add(Calendar.MONTH, -1 * rafOmru);
                break;
            case "YIL":
                cal.add(Calendar.YEAR, -1 * rafOmru);
                break;
            default:
                throw new MessageProcessCheckedException("Unexpected value: "
                        + rafOmruBirimi);
            }
            cal.add(Calendar.DATE, 1);
            resultTimeStamp = new Timestamp(cal.getTime().getTime());
        } else {
            final String msg = utsProductNumber
                    + " rafOmruBirimi or rafOmru cannot be empty.";
            throw new MessageProcessCheckedException(msg);
        }
        return resultTimeStamp;
    }

    private String searchMaterialUTS(int siteId, String utsProductNumber) {
        final String requestBody = "{\n\"UNO\" : \"" + utsProductNumber + "\"}";
        final String utsQueryUrl = "/rest/tibbiCihaz/urunSorgula";
        return sendUTSRestQueryRequest(siteId, requestBody, utsQueryUrl);

    }

    @Override
    @Nullable
    public String sendUTSRestQueryRequest(int siteId, String requestBody,
            String utsQueryUrl) {
        String utsUrl = env.getProperty("site" + siteId + ".uts_endpoint_url");
        String utsToken = env.getProperty("site" + siteId + ".uts_token");
        if (utsUrl != null && !"".equals(utsUrl) && utsToken != null && !""
                .equals(utsToken)) {
            OkHttpClient client = new OkHttpClient().newBuilder().build();
            MediaType mediaType = MediaType.parse(
                    javax.ws.rs.core.MediaType.TEXT_XML);
            RequestBody body = RequestBody.create(mediaType, requestBody);
            Request request = new Request.Builder().url(String.format("%s/%d%s",
```

```
                    utsUrl, siteId, utsQueryUrl)).method("POST", body)
                    .addHeader("utsToken", utsToken).addHeader("Content-Type",
                            javax.ws.rs.core.MediaType.TEXT_XML).build();

            try (Response response = client.newCall(request).execute()) {
                return response.body().string();
            } catch (Exception e) {
                LOGGER.error("Error occurred while querying UTS. ", e);
                return null;
            }
        } else {
            throw new SuUncheckedException(
                    "uts_endpoint_url or uts_token can not be empty.");
        }
    }

    private String getManufactureDateOfLotFromUTS(int siteId,
            String utsProductNumber, String lotNumber) {
        final String requestBody = String.format(
                "{\"LNO\" : \"%s\",\n\"UNO\" : \"%s\"}", lotNumber,
                utsProductNumber);
        final String utsQueryUrl = "/rest/ayrintiliTekilUrun/sorgula";
        String resStrProduct = sendUTSRestQueryRequest(siteId, requestBody,
                utsQueryUrl);
        if (!resStrProduct.contains("SNC")) {
            throw new IllegalStateException("Request to UTS server returned: "
                    + resStrProduct);
        } else {
            JSONObject jsonDetailedProductResult = new JSONObject(
                    resStrProduct);
            JSONArray detailedProductList = jsonDetailedProductResult
                    .getJSONArray("SNC");
            if (detailedProductList.length() != 0) {
                if (detailedProductList.length() > 1) {
                    LOGGER.error(String.format(
                            "/rest/ayrintiliTekilUrun/sorgula query result
returned MORE THAN ONE result for : %n %s",
                            requestBody));
                    return null;
                } else {
                    final Object manufactureDate = detailedProductList
                            .getJSONObject(0).get("uretimTarihiString");
                    if (!JSONObject.NULL.equals(manufactureDate)) {
                        String manufactureDateFormat = manufactureDate
                                .toString();
                        manufactureDateFormat += " 00:00:00";
                        return manufactureDateFormat;
                    } else {
                        LOGGER.error(String.format(
```

```
                                "/rest/ayrintiliTekilUrun/sorgula query result
returned NULL result for : %n %s",
                                requestBody));
                        return null;
                    }
                }
            } else {
                LOGGER.error(String.format(
                        "/rest/ayrintiliTekilUrun/sorgula query result returned
no result for : %n %s",
                        requestBody));
                return null;
            }
        }
    }

}
```