Overview

The EmmaTommy software executes an ETL procedure for extracting EMT's missions data from the Regional REST Service and load them in the local ERP.

Components

- EmmaHandler
  This service will cyclically download the EMT's missions data for the last 60 days, parse the received XML into N JSONS (one per mission) and send it over to the Converter layer

- EmmaTommyConverter
  This service receives as input a mission's JSON, unmarshals it into a Missione Object (which maps how the Regional Service views a mission), convert it into a Servizio Object (which maps how the ERP views a mission), marshals it into a JSON and sends it over to the DataHandler

- TommyDataHandler
  This service will receives Servizio Objects as JSON, will check if it is to be posted or not based upon the information received from the Dbs and will eventually save it into the Staging DB

- TommyPoster
  This service will cyclically iterate over the Staging DB to find new missions to be posted to the local ERP via REST. It will also handle the posting output and react accordingly, moving the missions to the Persistence DB or to the Error Section of the Staging DB

- Staging DB
  This service manages the DB where the missions ready to be posted are temporally stored.

- Persistence DB
  This service manages the DB where the already posted missions are stored.

- Analitics
  This service is linked to the Persistence DB, to the Logs and to the Servers.
  It will compute statistics, visualizations and will react to events by sending alerts.

Tech

The system is written using Java 11 as an ensemble of micro-services. Internally micro-services are composed by concurrent Actors via AKKA. At the global communication level between microservices, Akka is used to ask for data, for application logic or for orchestration, while Kafka is used a persistent data channel for producing micro-services' output that other micro-services react on.
The chosen staging DB is MySQL, where the data element is a decoration of a Servizio's JSON to quickly analyze them for posting.
The persistence DB instead is MongoDB, where the JSONs are stored raw without any decoration.
Finally, the analytics service uses ELK stack linked to each micro-services' logs (via FileBeat and LogStash) and to the persistence DB (via LogStash and Kafka, to add persistence to the data link).

Extract Workflow

EmmaHandler
EmmaHandler downloads new XML with N Missioni
EmmaHandler will parse the XML into N JSON Missioni
EmmaHandler will send the N JSON Missioni over to Kafka

Transform Workflow

EmmaTommyConverter
EmmaTommyConverter will consume a new JSON Missione from Kafka
EmmaTommyConverter will convert the JSON Missione to a JSON Servizio
EmmaTommyConverter will send the JSON Servizio over to Kafka

TommyHandler
TommyHandler will consume a new JSON Servizio from to Kafka
TommyHandler will determine what to do with the new Servizio based on the following cases

- If the servizio (by ID) is already present in the *Persistence DB*
  - If the servizio has the same data (no update), discard it
  - If the servizio has updates, send error notification for manual update
- If the servizio (by ID) is not present in the *Persistence DB*
  - If the servizio (by ID) is already present in the *Staging DB error section*
    - If the servizio has the same data (no update), discard it
    - If the servizio has updates, update the database entry and move it to the new section
  - If the servizio (by ID) is not present in the *Staging DB error section*
    - If the servizio (by ID) is present in the Staging DB new section
      - If the servizio has the same data (no update), discard it
      - If the servizio has updates, update the database entry
    - If the servizio (by ID) is not present in the Staging DB new section
      - Decorate the servizio and create a new DB entry for it

Load Workflow

TommyPoster (Ciclically every X time)
Lock the *Persistence DB* (if it fails, wait till it can be locked with timeout)
Lock the *Staging DB* (if it fails, unlock the *Persistence DB* and abort)
For all the DB entries in the *Error Section*

- Set the MEZZO_ID as blocked for posting.

For all the DB entries in the New *Section* (Analysis ordered by servizio ID)

- Check if the Servizio is valid:
    - KM > 0
    - It has to have a MEZZO_ID
    - The MEZZO_ID must not be blocked for posting
    - The servizio should not be also present in the *Error Section* (Double Checking)
    - The servizio should not be also present in the *Persistence DB* (Double Checking)
- If the servizio is valid add it to the list of servizi to be posted (classified by MEZZO_ID)
- If the servizio is not valid, set the MEZZO_ID as blocked for posting

For every MEZZO_ID list (if not empty)

- If there isn't a poster for that MEZZO_ID, spawn it
- Send the list to the poster for that MEZZO_ID
- Wait for the Poster Response, which can be of three types:
    - Unable to Send Over to REST
    - Sent Over to REST
        - REST accepted the data
            - Send eventual received warnings for manual corrections
            - Write all the posted servizi in to the *Persistence DB*
            - Remove all the posted servizi from the *Staging DB* New *Section*
        - REST refuted the data with error
          (the whole list of servizi is not posted if at least one servizio generates an error)
            - Move all the the unposted servizi from the *Staging DB* New *Section* to the *Error Section*

Unlock the *Persistence DB*
Unlock the *Staging DB*