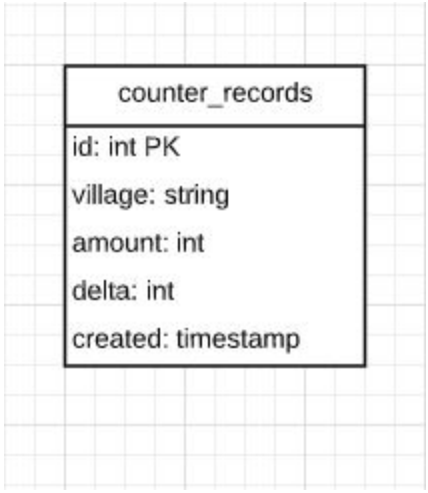
MODEL

- I've decided to create only one table for this task.



Id - primary key for this table.
Village - village name.
Amount - current counter number.
Delta - amount minus previous amount.
Created - date and time of counter record insert. Purpose of this field is to provide a possibility to find a lates counter record before inserting a new one.

- Table independent b-tree indexes for fields id, village, created.

SYSTEM ARCHITECTURE

- This application consists of simple REST service which saves counter record in db and later can generate reports.
- I've decided to call counters service for village name before saving counter records. This solution has one downside. Rest calls is not the most stable way of communication, so from time to time 4xx or 5xx responses can be obtained due to different reasons. More reliable solution will to to call counter service in separate job and save data in corresponding table. Such table would consist of counter id to village name mapping. After that during consumption report degeneration counter records and village name tables should be joined. In this case counter records table has couner_id fields instead of village. Of course, such report can ignored counters which don't have counter to village mapping yet. But current implementation contains easiest solution.
- Also, I add concurrent counter record save solution. But where to use it or not also depends on counter service implementation.
- I've decided to make duration report parameter not only 24h but from 1h to 100h.
- There is no validation if previous amount of consumed energy is lower than the next one before insert. Theoretically such situation is possible with real counters. Also there is no validation if it is less or greater than zero.
- Application lacks database evolution tool like flyway or liquibase.

- Logging is done only to system out, but with slf4j interface. So, logs should be properly configured.
- Project contains TODOs in places which should be improved.

TECHNOLOGY AND FRAMEWORKS CHOICE

- This service is Spring Boot application.
- Spring Data JPA for connectivity with database, because it is simple in use.
- Lombok for convenience in work with POJO.
- Spring MVC for REST services.
- Feign for creating declarative REST clients.

TESTING STRATEGY

- This project contains unit and integration tests, which is enough for main functionality testing.
- Smoke manual testing.
- But this project lack load. Such kind of test should be done on real environments and with real database, not in memory.
- This application has integration with third party services. Which submit counter records and contains additional counters data, like village name. Because API can evolve with time, it is obvious it is potentially unstable place in the application.