Рударење на масивни податоци
Домашна задача 1

Даријан Шекеров – 216055

Јава проект – Flink

Најпрво се преземаат отчитувањата од сензорите така што правиме Kafka consumer кој има десеријализатор кој прави објект од JSON и го додава timestamp-от потребен за временските прозорци.
Требаше да се додаде `env.setParallelism(1);` за да работат прозорците со timestamp. Се креира source со потребен watermark strategy.
Во `result1Stream` се прави

- `keyBy` за да се групираат записите според клучот
- `window` со `SlidingEventTimeWindow` со големина 5с и поместување 1с
- `process` каде што за секој клуч се брои колку записи има со таков клуч и се враќа стринг кој кажува дека за тој клуч, во прозорецот со тие граници се изброени толку записи
- На крај тој стрим се праќа назад во Kafka така што се отвора sink. Овој синк работи со стрингови, па затоа се користи `SimpleStringSchema`

Во `result2Stream` се прави

- `keyBy` за да се групираат записите според клучот
- `window` со `SlidingEventTimeWindow` со големина 5с и поместување 1с
- `process` каде што за секој клуч се прави `IntSummaryStatistics` кој пресметува статистики за сите записи во прозорецот со тој клуч. Тука наместо стринг се враќа објект `SensorDataStatistics`.
- На крај тој стрим се праќа назад во Kafka така што се отвора sink. Овој синк работи со `SensorDataStatistics`, па затоа се користи серијализатор кој го претвора објектот во JSON

На крај се повикува `env.execute` за да се изврши таскот на Flink.

Python скрипти

`Produce_messages.py` остана горе долу иста, само се тргна timestamp полето бидејќи самиот Kafka додава timestamp кога се праќа пораката.
`Consume_messages.py` е променета за да работи со повеќе topics одеднаш така што со `subscribe` ги праќаме topics-от за кои сме заинтересирани и после со `poll` проверуваме дали има нови пораки и ако има ги печатиме за секој topic посебно.