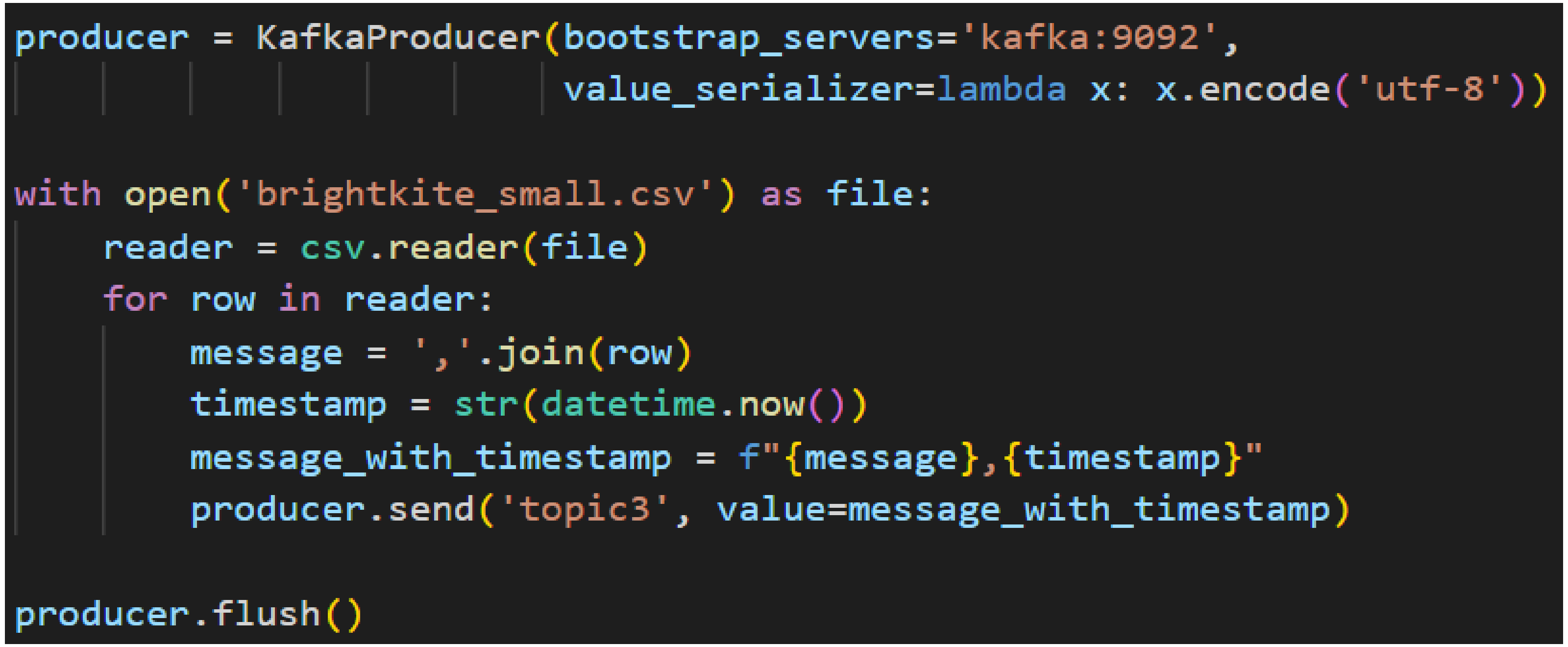
Projekat 2 – Big data

Nikola Đorđević 1567

Kafka producer

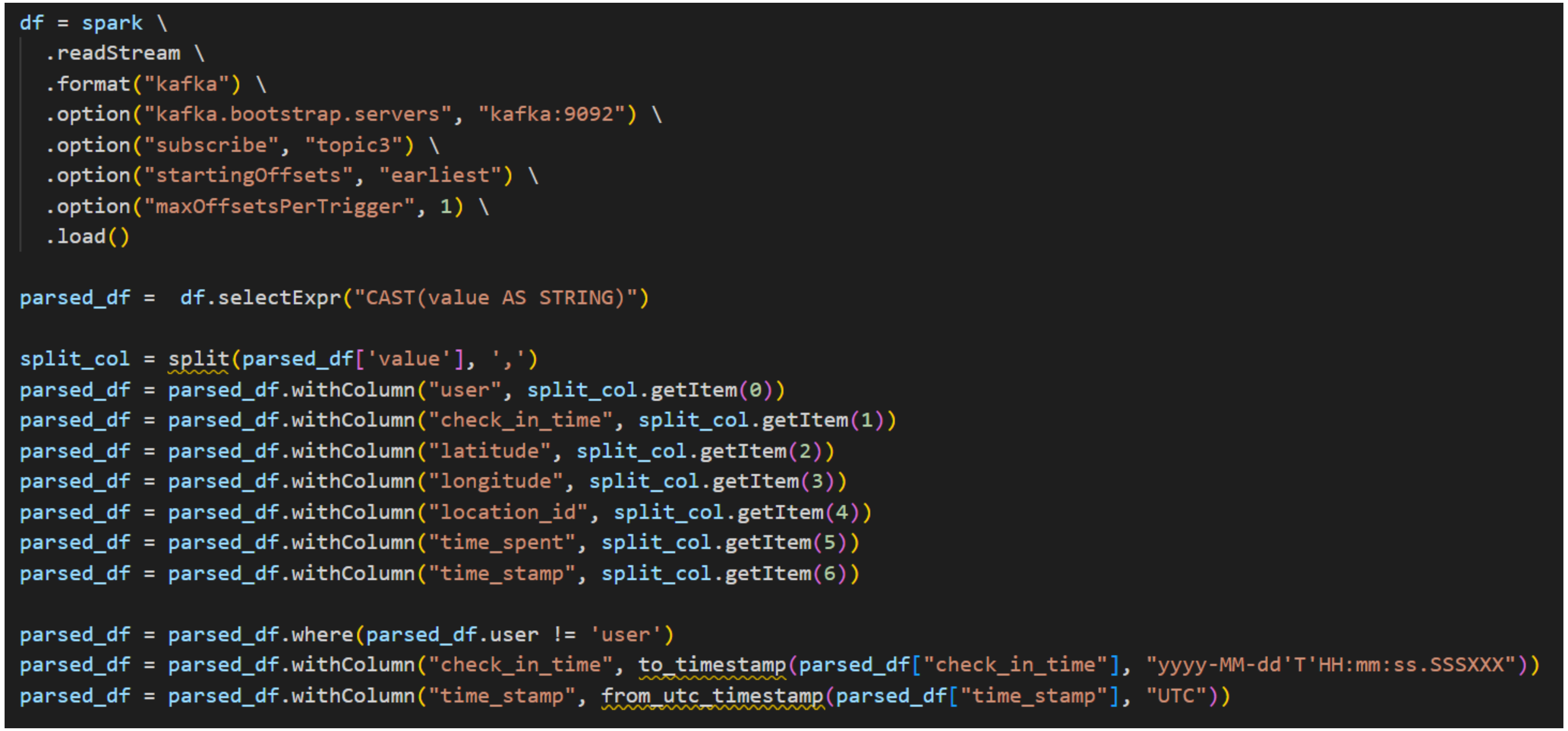


```
producer = KafkaProducer(bootstrap_servers='kafka:9092',
                         value_serializer=lambda x: x.encode('utf-8'))

with open('brightkite_small.csv') as file:
    reader = csv.reader(file)
    for row in reader:
        message = ','.join(row)
        timestamp = str(datetime.now())
        message_with_timestamp = f"{message},{timestamp}"
        producer.send('topic3', value=message_with_timestamp)

producer.flush()
```

- Kafka producer čita red po red iz fajla u kome se nalaze podaci, i pročitani red šalje na odgovarajući topic
- Polja jednog reda se šalju odvojena zarezom
- Za svaki poslati red se dodaje i vremenska oznaka kada je poslat

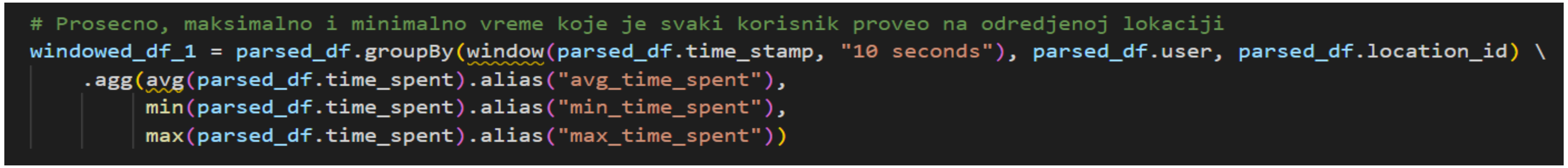
Spark streaming aplikacija

Čitanje i parsiranje podataka sa Kafka topic-a

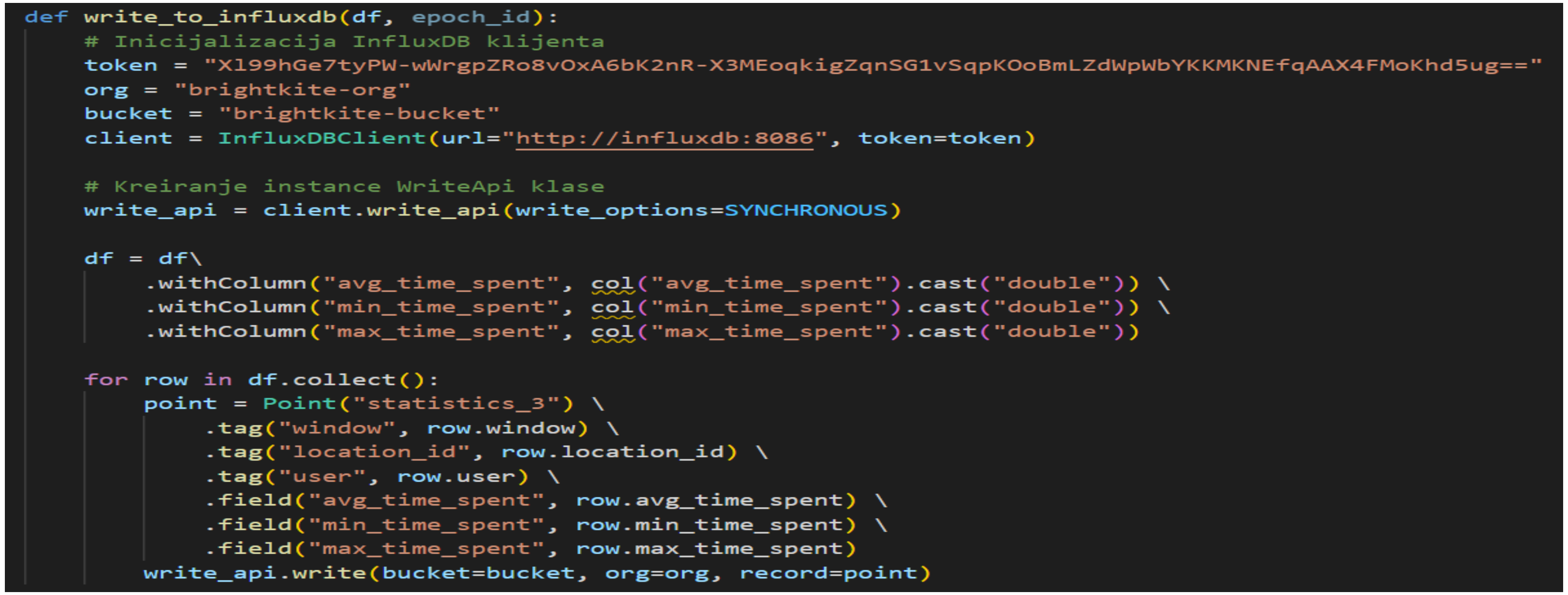


```
df = spark \
  .readStream \
  .format("kafka") \
  .option("kafka.bootstrap.servers", "kafka:9092") \
  .option("subscribe", "topic3") \
  .option("startingOffsets", "earliest") \
  .option("maxOffsetsPerTrigger", 1) \
  .load()

parsed_df =  df.selectExpr("CAST(value AS STRING)")

split_col = split(parsed_df['value'], ',')
parsed_df = parsed_df.withColumn("user", split_col.getItem(0))
parsed_df = parsed_df.withColumn("check_in_time", split_col.getItem(1))
parsed_df = parsed_df.withColumn("latitude", split_col.getItem(2))
parsed_df = parsed_df.withColumn("longitude", split_col.getItem(3))
parsed_df = parsed_df.withColumn("location_id", split_col.getItem(4))
parsed_df = parsed_df.withColumn("time_spent", split_col.getItem(5))
parsed_df = parsed_df.withColumn("time_stamp", split_col.getItem(6))

parsed_df = parsed_df.where(parsed_df.user != 'user')
parsed_df = parsed_df.withColumn("check_in_time", to_timestamp(parsed_df["check_in_time"], "yyyy-MM-dd'T'HH:mm:ss.SSSXXX"))
parsed_df = parsed_df.withColumn("time_stamp", from_utc_timestamp(parsed_df["time_stamp"], "UTC"))
```

Statistiški proračuni za provedeno vreme na određenoj lokaciji, za određenog korisnika



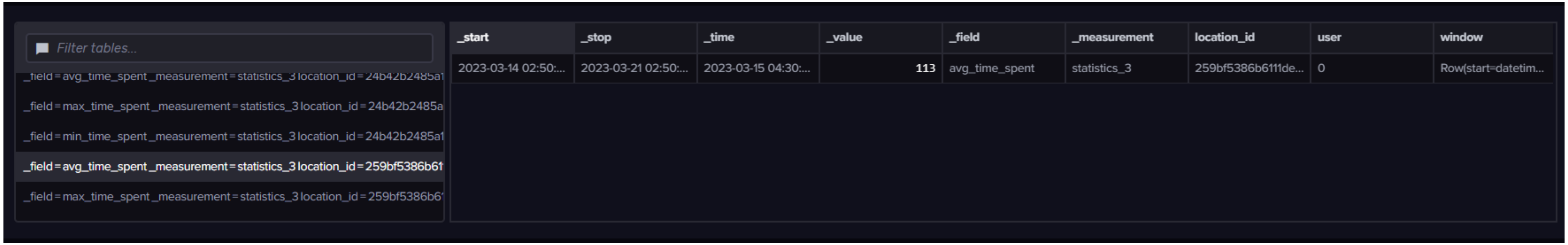
```python
# Prosecno, maksimalno i minimalno vreme koje je svaki korisnik proveo na odredjenoj lokaciji
windowed_df_1 = parsed_df.groupBy(window(parsed_df.time_stamp, "10 seconds"), parsed_df.user, parsed_df.location_id) \
    .agg(avg(parsed_df.time_spent).alias("avg_time_spent"),
         min(parsed_df.time_spent).alias("min_time_spent"),
         max(parsed_df.time_spent).alias("max_time_spent"))
```



```python
def write_to_influxdb(df, epoch_id):
    # Inicijalizacija InfluxDB klijenta
    token = "Xl99hGe7tyPW-wWrgpZRo8vOxA6bK2nR-X3MEoqkigZqnSG1vSqpKOoBmLZdWpWbYKKMKNEfqAAX4FMoKhd5ug=="
    org = "brightkite-org"
    bucket = "brightkite-bucket"
    client = InfluxDBClient(url="http://influxdb:8086", token=token)

    # Kreiranje instance WriteApi klase
    write_api = client.write_api(write_options=SYNCHRONOUS)

    df = df\
        .withColumn("avg_time_spent", col("avg_time_spent").cast("double")) \
        .withColumn("min_time_spent", col("min_time_spent").cast("double")) \
        .withColumn("max_time_spent", col("max_time_spent").cast("double"))

    for row in df.collect():
        point = Point("statistics_3") \
            .tag("window", row.window) \
            .tag("location_id", row.location_id) \
            .tag("user", row.user) \
            .field("avg_time_spent", row.avg_time_spent) \
            .field("min_time_spent", row.min_time_spent) \
            .field("max_time_spent", row.max_time_spent)
        write_api.write(bucket=bucket, org=org, record=point)
```

- Za vremenski prozor od 10 sekundi, podaci se grupišu po korisnicima i lokacijama
- Računa se prosečno, minimalno i maksimalno vreme koje je korisnik proveo na lokaciji

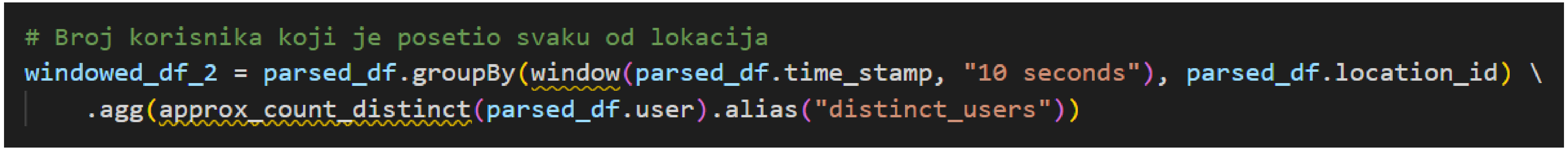
Rezultat u InfluxDB bazi podataka



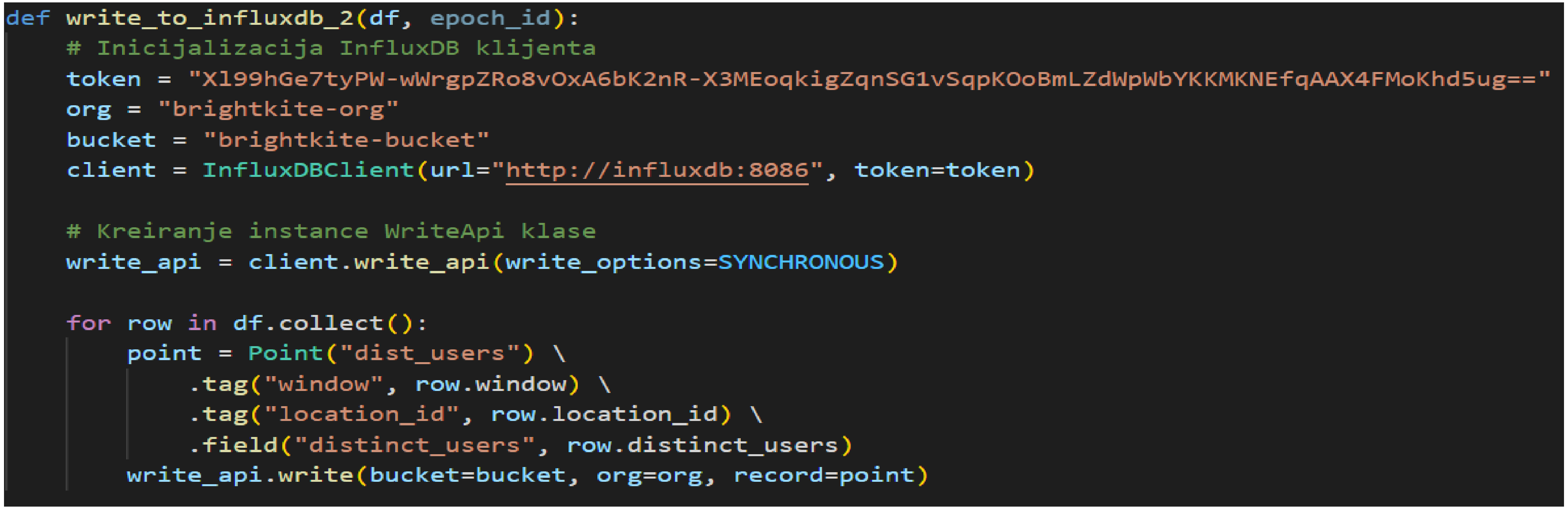
```
_field = avg_time_spent _measurement = statistics_3 location_id = 24b42b2485a1
_field = max_time_spent _measurement = statistics_3 location_id = 24b42b2485a
_field = min_time_spent _measurement = statistics_3 location_id = 24b42b2485a1
_field = avg_time_spent _measurement = statistics_3 location_id = 259bf5386b61
_field = max_time_spent _measurement = statistics_3 location_id = 259bf5386b61
```

_start	_stop	_time	_value	_field	_measurement	location_id	user	window
2023-03-14 02:50:...	2023-03-21 02:50:...	2023-03-15 04:30:...	113	avg_time_spent	statistics_3	259bf5386b6111de...	0	Row(start=datetim...

Broj korisnika koji je posetio svaku od lokacija



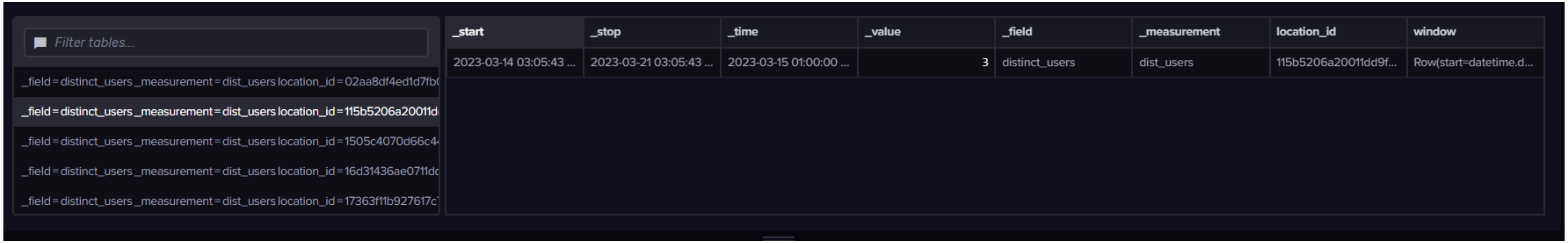
```
# Broj korisnika koji je posetio svaku od lokacija
windowed_df_2 = parsed_df.groupBy(window(parsed_df.time_stamp, "10 seconds"), parsed_df.location_id) \
    .agg(approx_count_distinct(parsed_df.user).alias("distinct_users"))
```



```
def write_to_influxdb_2(df, epoch_id):
    # Inicijalizacija InfluxDB klijenta
    token = "Xl99hGe7tyPW-wWrgpZRo8vOxA6bK2nR-X3MEoqkigZqnSG1vSqpKOoBmLZdWpWbYKKMKNEfqAAX4FMoKhd5ug=="
    org = "brightkite-org"
    bucket = "brightkite-bucket"
    client = InfluxDBClient(url="http://influxdb:8086", token=token)

    # Kreiranje instance WriteApi klase
    write_api = client.write_api(write_options=SYNCHRONOUS)

    for row in df.collect():
        point = Point("dist_users") \
            .tag("window", row.window) \
            .tag("location_id", row.location_id) \
            .field("distinct_users", row.distinct_users)
        write_api.write(bucket=bucket, org=org, record=point)
```

- Određivanje broja korisnika koji je posetio svaku od lokacija za vremenski prozor od 10 sekundi, i upis rezultata u InfluxDB bazu podataka

Rezultat u InfluxDB bazi podataka



_field = distinct_users _measurement = dist_users location_id = 02aa8df4ed1d7fb

_field = distinct_users _measurement = dist_users location_id = 115b5206a20011d

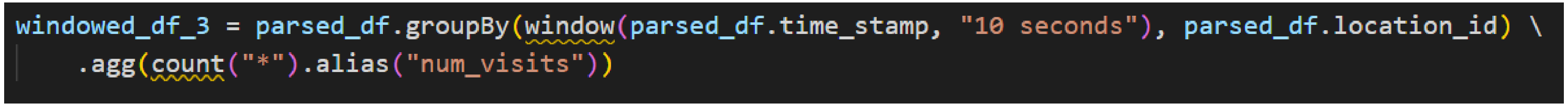
_field = distinct_users _measurement = dist_users location_id = 1505c4070d66c4

_field = distinct_users _measurement = dist_users location_id = 16d31436ae0711d

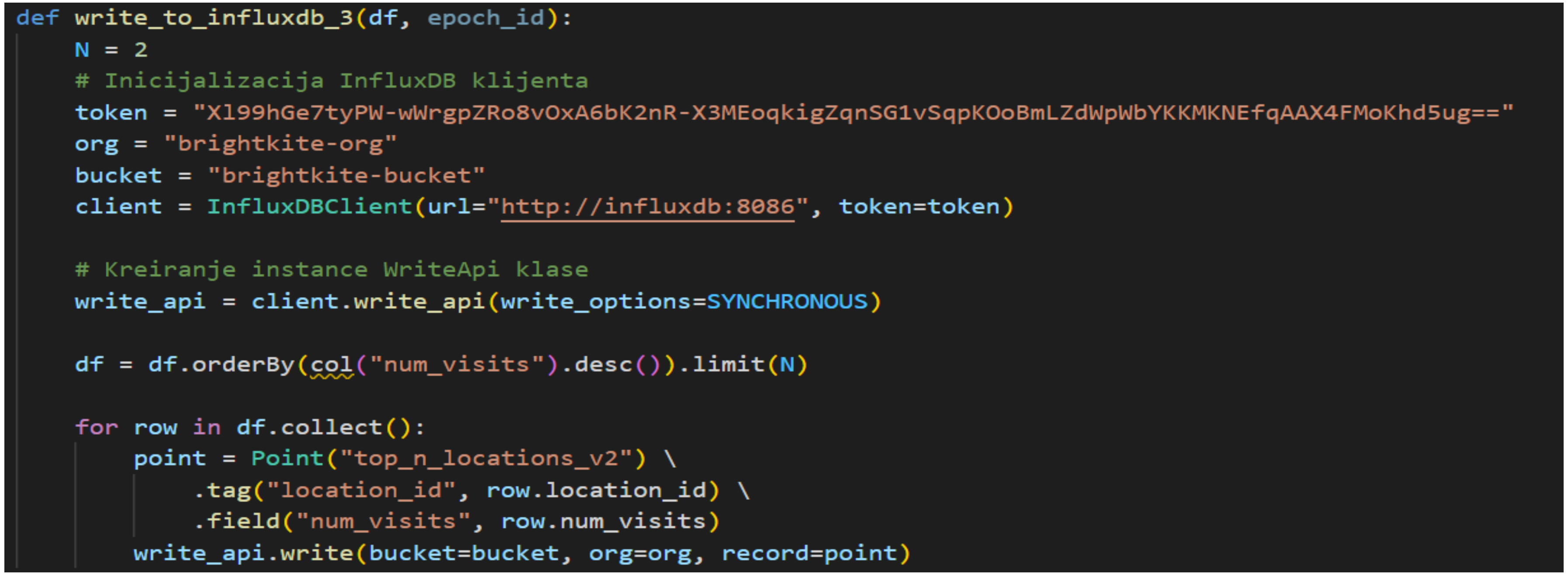
_field = distinct_users _measurement = dist_users location_id = 17363f11b927617c

_start	_stop	_time	_value	_field	_measurement	location_id	window
2023-03-14 03:05:43 ...	2023-03-21 03:05:43 ...	2023-03-15 01:00:00 ...	3	distinct_users	dist_users	115b5206a20011dd9f...	Row(start=datetime.d...

Određivanje top N lokacija i upis u InfluxDB



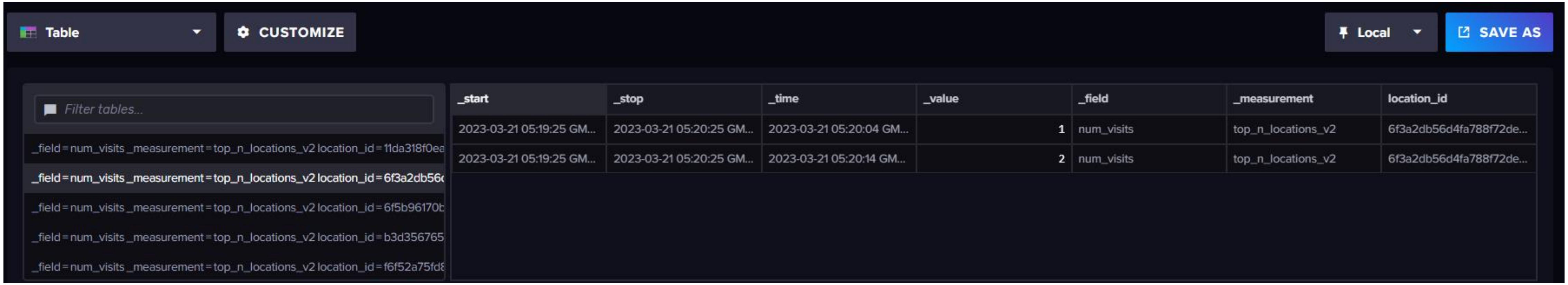
```python
windowed_df_3 = parsed_df.groupBy(window(parsed_df.time_stamp, "10 seconds"), parsed_df.location_id) \
    .agg(count("*").alias("num_visits"))
```



```python
def write_to_influxdb_3(df, epoch_id):
    N = 2
    # Inicijalizacija InfluxDB klijenta
    token = "Xl99hGe7tyPW-wWrgpZRo8vOxA6bK2nR-X3MEoqkigZqnSG1vSqpKOoBmLZdWpWbYKKMKNEfqAAX4FMoKhd5ug=="
    org = "brightkite-org"
    bucket = "brightkite-bucket"
    client = InfluxDBClient(url="http://influxdb:8086", token=token)

    # Kreiranje instance WriteApi klase
    write_api = client.write_api(write_options=SYNCHRONOUS)

    df = df.orderBy(col("num_visits").desc()).limit(N)

    for row in df.collect():
        point = Point("top_n_locations_v2") \
            .tag("location_id", row.location_id) \
            .field("num_visits", row.num_visits)
        write_api.write(bucket=bucket, org=org, record=point)
```

Na osnovu broja poseta za određenu lokaciju, u vremenskom prozoru od 10 sekundi, vrši se sortiranje lokacija i određivanje top N lokacija, koje se upisuju u InfluxDB bazu

Rezultat u InfluxDB bazi podataka



_field = num_visits _measurement = top_n_locations_v2 location_id = 11da318f0ea

_field = num_visits _measurement = top_n_locations_v2 location_id = 6f3a2db56c

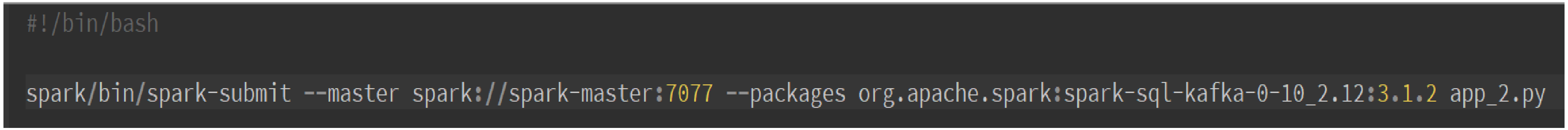
_field = num_visits _measurement = top_n_locations_v2 location_id = 6f5b96170b

_field = num_visits _measurement = top_n_locations_v2 location_id = b3d356765

_field = num_visits _measurement = top_n_locations_v2 location_id = f6f52a75fd8

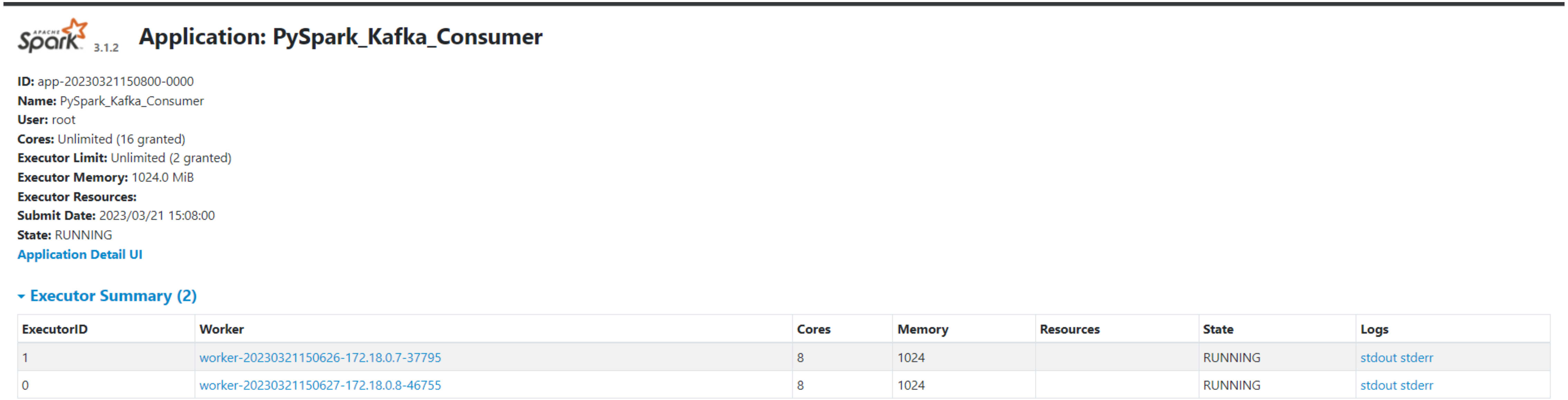
_start	_stop	_time	_value	_field	_measurement	location_id
2023-03-21 05:19:25 GM...	2023-03-21 05:20:25 GM...	2023-03-21 05:20:04 GM...	1	num_visits	top_n_locations_v2	6f3a2db56d4fa788f72de...
2023-03-21 05:19:25 GM...	2023-03-21 05:20:25 GM...	2023-03-21 05:20:14 GM...	2	num_visits	top_n_locations_v2	6f3a2db56d4fa788f72de...

Skripta za pokretanje aplikacije



```
#!/bin/bash

spark/bin/spark-submit --master spark://spark-master:7077 --packages org.apache.spark:spark-sql-kafka-0-10_2.12:3.1.2 app_2.py
```

Web UI



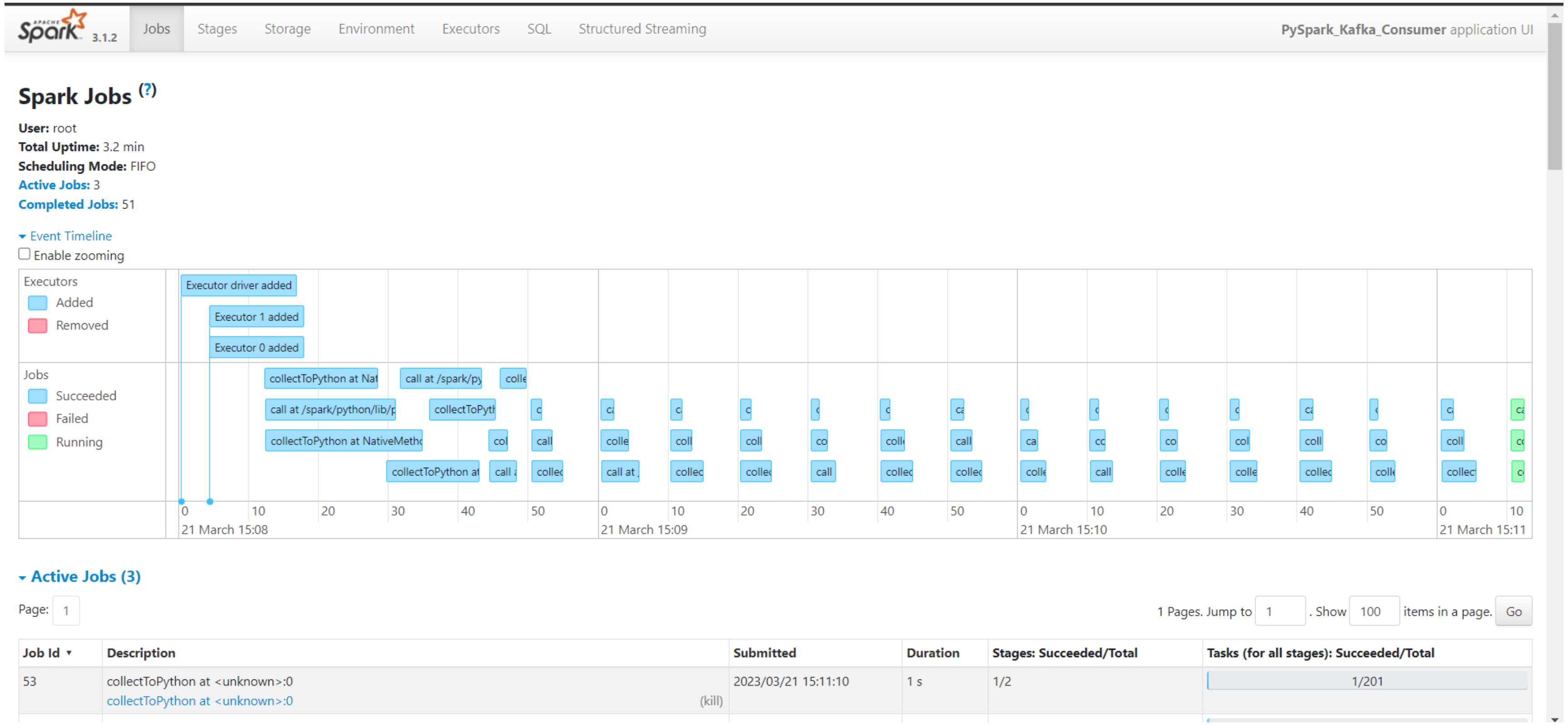
Application: PySpark_Kafka_Consumer

ID: app-20230321150800-0000
Name: PySpark_Kafka_Consumer
User: root
Cores: Unlimited (16 granted)
Executor Limit: Unlimited (2 granted)
Executor Memory: 1024.0 MiB
Executor Resources:
Submit Date: 2023/03/21 15:08:00
State: RUNNING
Application Detail UI

Executor Summary (2)

ExecutorID	Worker	Cores	Memory	Resources	State	Logs
1	worker-20230321150626-172.18.0.7-37795	8	1024		RUNNING	stdout stderr
0	worker-20230321150627-172.18.0.8-46755	8	1024		RUNNING	stdout stderr

Web UI



Spark 3.1.2 | Jobs | Stages | Storage | Environment | Executors | SQL | Structured Streaming | **PySpark_Kafka_Consumer** application UI

Spark Jobs (?)

User: root
Total Uptime: 3.2 min
Scheduling Mode: FIFO
Active Jobs: 3
Completed Jobs: 51

▾ Event Timeline
□ Enable zooming

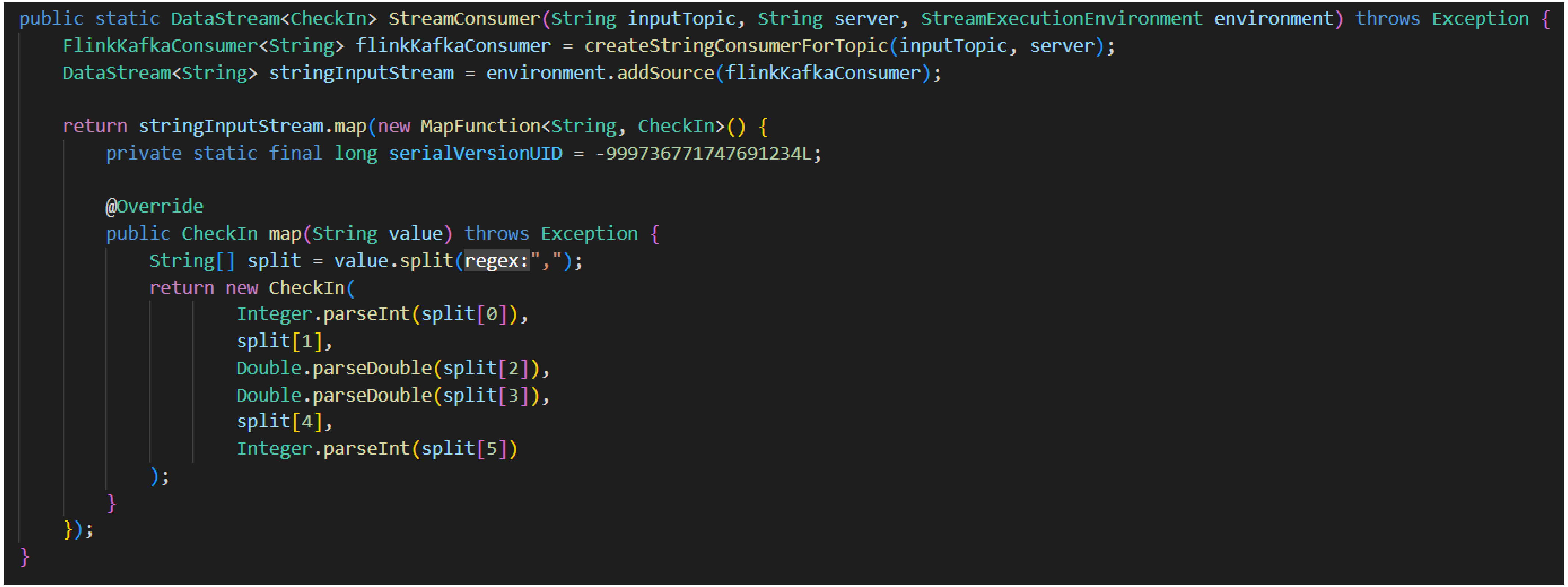
▾ **Active Jobs (3)**

Page: 1 | 1 Pages. Jump to 1 . Show 100 items in a page. Go

Job Id ▾	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
53	collectToPython at <unknown>:0 collectToPython at <unknown>:0 (kill)	2023/03/21 15:11:10	1 s	1/2	1/201

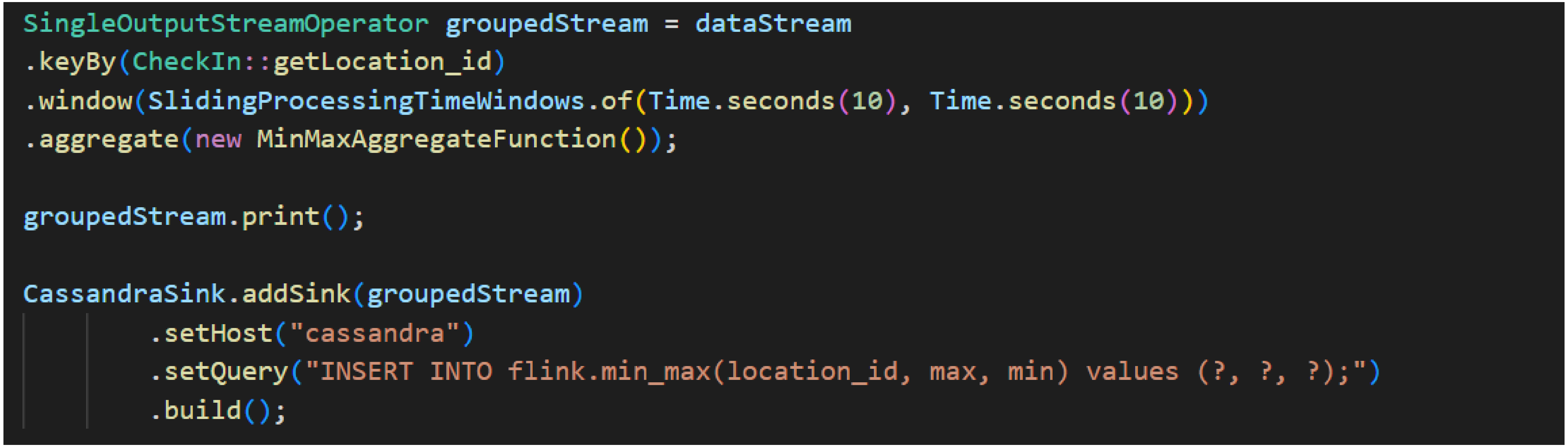
Flink aplikacija

Čitanje i parsiranje podataka sa Kafka topic-a



```java
public static DataStream<CheckIn> StreamConsumer(String inputTopic, String server, StreamExecutionEnvironment environment) throws Exception {
    FlinkKafkaConsumer<String> flinkKafkaConsumer = createStringConsumerForTopic(inputTopic, server);
    DataStream<String> stringInputStream = environment.addSource(flinkKafkaConsumer);

    return stringInputStream.map(new MapFunction<String, CheckIn>() {
        private static final long serialVersionUID = -999736771747691234L;

        @Override
        public CheckIn map(String value) throws Exception {
            String[] split = value.split(regex:",");
            return new CheckIn(
                    Integer.parseInt(split[0]),
                    split[1],
                    Double.parseDouble(split[2]),
                    Double.parseDouble(split[3]),
                    split[4],
                    Integer.parseInt(split[5])
            );
        }
    });
}
```

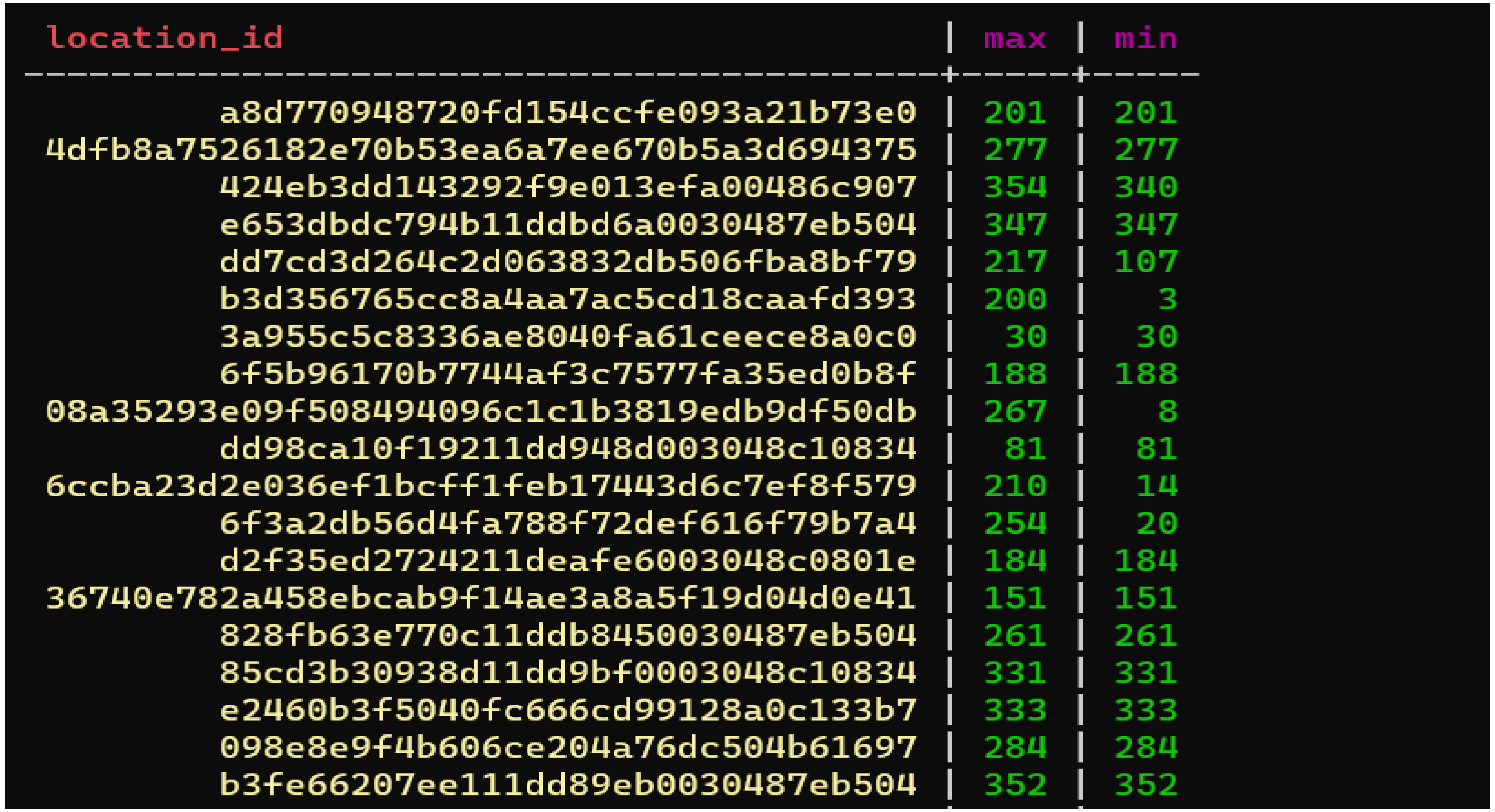
Statistički proračuni za provedeno vreme na određenoj lokaciji



```java
SingleOutputStreamOperator groupedStream = dataStream
.keyBy(CheckIn::getLocation_id)
.window(SlidingProcessingTimeWindows.of(Time.seconds(10), Time.seconds(10)))
.aggregate(new MinMaxAggregateFunction());

groupedStream.print();

CassandraSink.addSink(groupedStream)
        .setHost("cassandra")
        .setQuery("INSERT INTO flink.min_max(location_id, max, min) values (?, ?, ?);")
        .build();
```

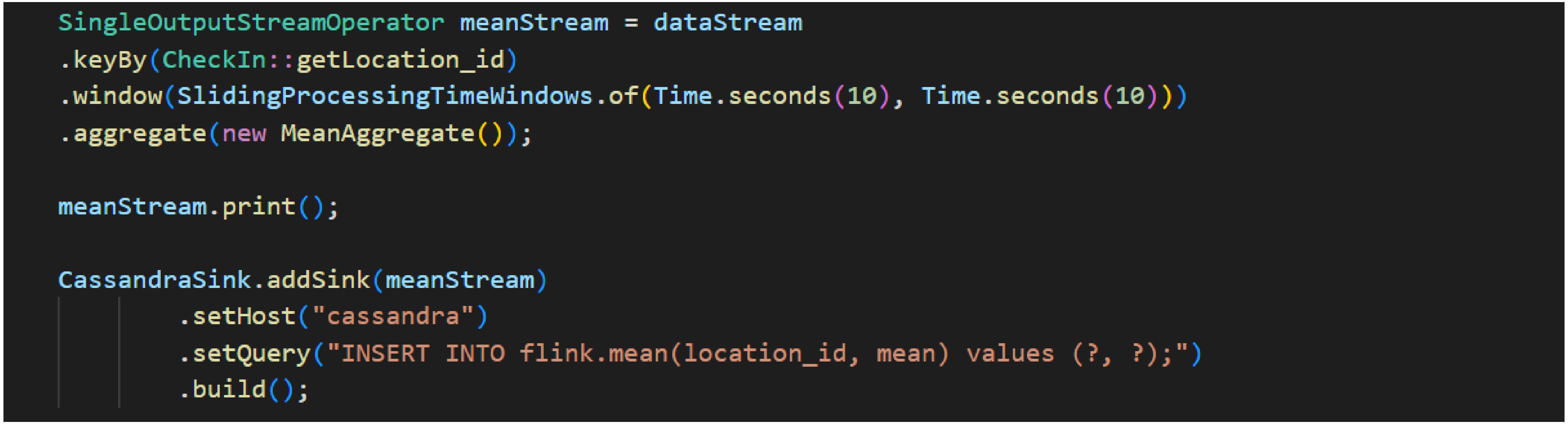
- Određivanje minimalnog i maksimalnog provedenog vremena na svakoj od lokacija i upis rezultata u Cassandra bazu podataka
- Podaci su grupisani u vremenski (po tipu "Sliding") prozor veličine 10 sekundi i koraka 10 sekundi

Rezultat u Cassandra DB



```
 location_id                              | max | min
------------------------------------------+-----+-----
         a8d770948720fd154ccfe093a21b73e0 | 201 | 201
 4dfb8a7526182e70b53ea6a7ee670b5a3d694375 | 277 | 277
         424eb3dd143292f9e013efa00486c907 | 354 | 340
         e653dbdc794b11ddbd6a0030487eb504 | 347 | 347
         dd7cd3d264c2d063832db506fba8bf79 | 217 | 107
         b3d356765cc8a4aa7ac5cd18caafd393 | 200 |   3
         3a955c5c8336ae8040fa61ceece8a0c0 |  30 |  30
         6f5b96170b7744af3c7577fa35ed0b8f | 188 | 188
 08a35293e09f508494096c1c1b3819edb9df50db | 267 |   8
         dd98ca10f19211dd948d003048c10834 |  81 |  81
 6ccba23d2e036ef1bcff1feb17443d6c7ef8f579 | 210 |  14
         6f3a2db56d4fa788f72def616f79b7a4 | 254 |  20
         d2f35ed2724211deafe6003048c0801e | 184 | 184
 36740e782a458ebcab9f14ae3a8a5f19d04d0e41 | 151 | 151
         828fb63e770c11ddb8450030487eb504 | 261 | 261
         85cd3b30938d11dd9bf0003048c10834 | 331 | 331
         e2460b3f5040fc666cd99128a0c133b7 | 333 | 333
         098e8e9f4b606ce204a76dc504b61697 | 284 | 284
         b3fe66207ee111dd89eb0030487eb504 | 352 | 352
```

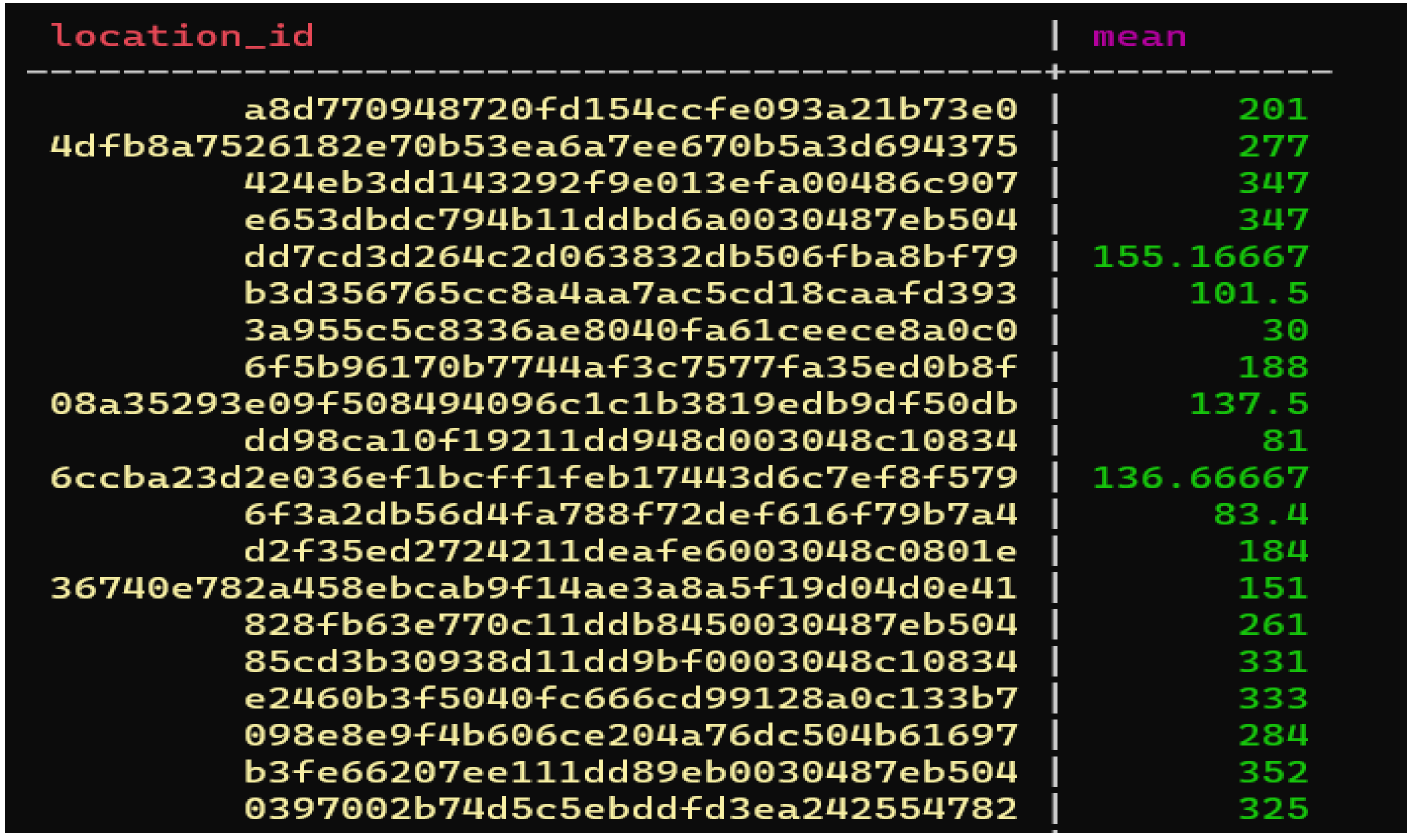
Statistički proračuni za provedeno vreme na određenoj lokaciji



```
SingleOutputStreamOperator meanStream = dataStream
.keyBy(CheckIn::getLocation_id)
.window(SlidingProcessingTimeWindows.of(Time.seconds(10), Time.seconds(10)))
.aggregate(new MeanAggregate());

meanStream.print();

CassandraSink.addSink(meanStream)
        .setHost("cassandra")
        .setQuery("INSERT INTO flink.mean(location_id, mean) values (?, ?);")
        .build();
```

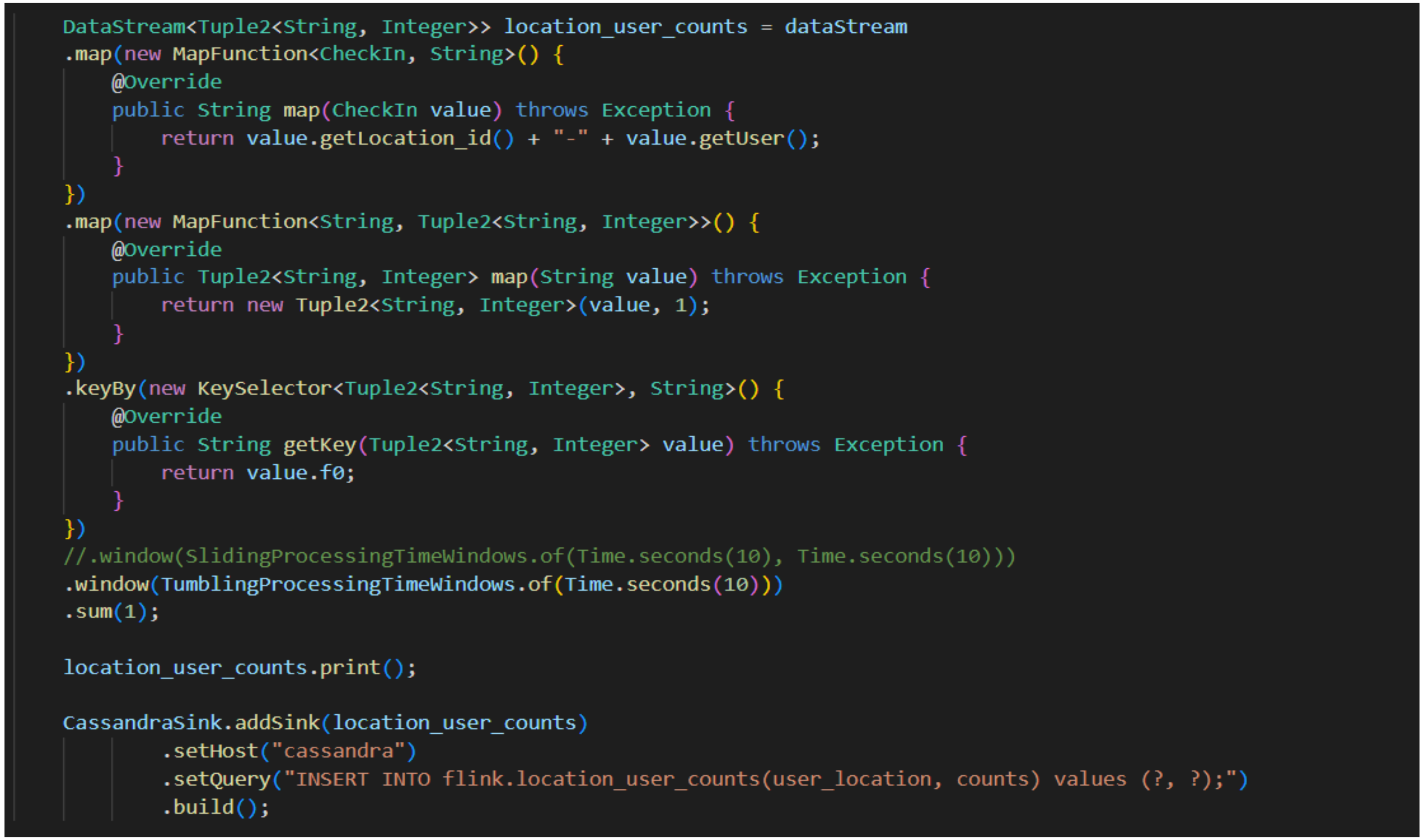
- Određivanje prosečnog provedenog vremena na svakoj od lokacija i upis rezultata u Cassandra bazu podataka
- Podaci su grupisani u vremenski (po tipu "Sliding") prozor veličine 10 sekundi i koraka 10 sekundi

Rezultat u Cassandra DB



```
 location_id                              | mean
------------------------------------------+-----------
         a8d770948720fd154ccfe093a21b73e0 |       201
 4dfb8a7526182e70b53ea6a7ee670b5a3d694375 |       277
         424eb3dd143292f9e013efa00486c907 |       347
         e653dbdc794b11ddbd6a0030487eb504 |       347
         dd7cd3d264c2d063832db506fba8bf79 | 155.16667
         b3d356765cc8a4aa7ac5cd18caafd393 |     101.5
         3a955c5c8336ae8040fa61ceece8a0c0 |        30
         6f5b96170b7744af3c7577fa35ed0b8f |       188
 08a35293e09f508494096c1c1b3819edb9df50db |     137.5
         dd98ca10f19211dd948d003048c10834 |        81
 6ccba23d2e036ef1bcff1feb17443d6c7ef8f579 | 136.66667
         6f3a2db56d4fa788f72def616f79b7a4 |      83.4
         d2f35ed2724211deafe6003048c0801e |       184
 36740e782a458ebcab9f14ae3a8a5f19d04d0e41 |       151
         828fb63e770c11ddb8450030487eb504 |       261
         85cd3b30938d11dd9bf0003048c10834 |       331
         e2460b3f5040fc666cd99128a0c133b7 |       333
         098e8e9f4b606ce204a76dc504b61697 |       284
         b3fe66207ee111dd89eb0030487eb504 |       352
         0397002b74d5c5ebddfd3ea242554782 |       325
```

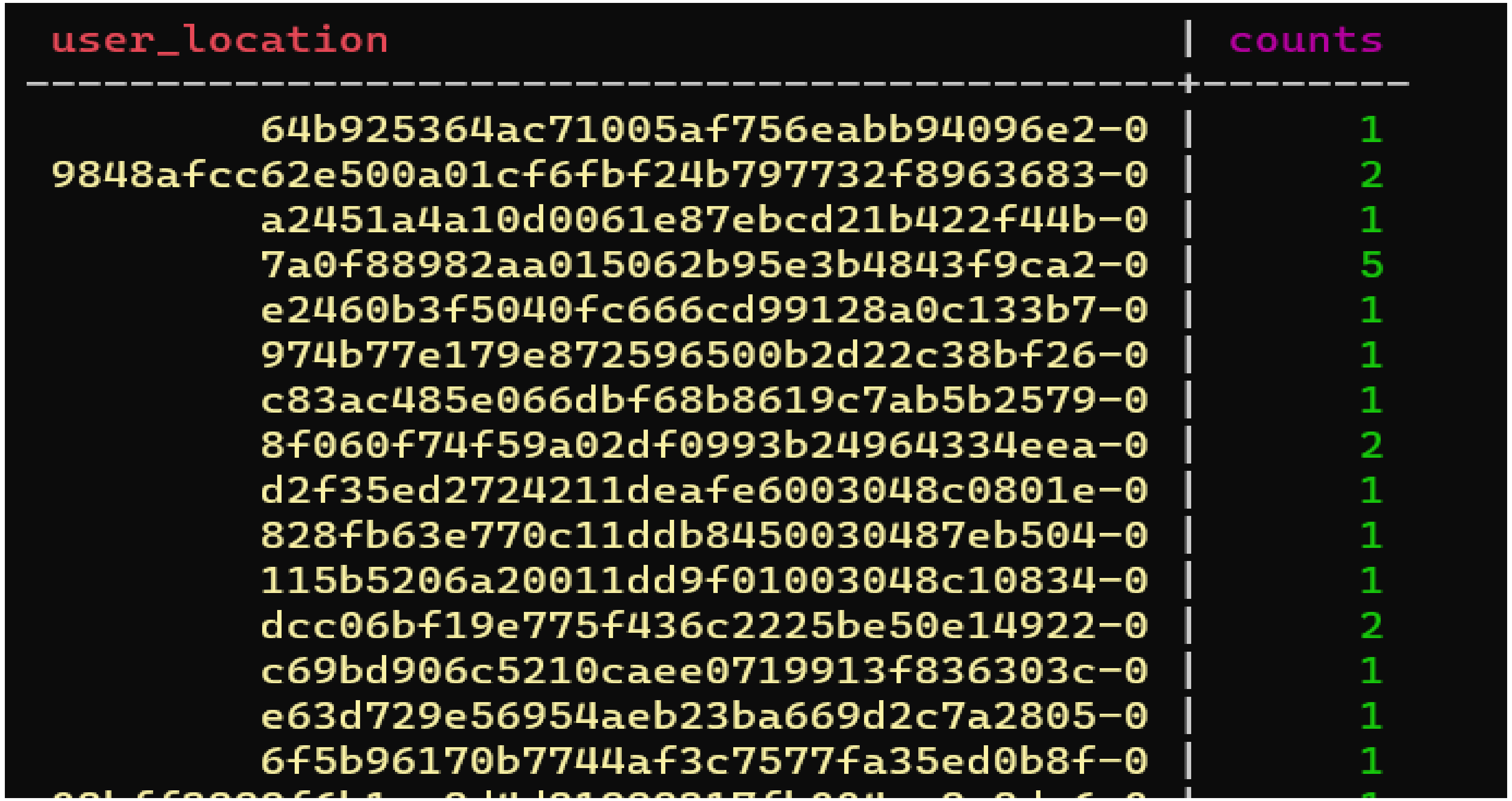
Broj korisnika koji je posetio svaku od lokacija



```
DataStream<Tuple2<String, Integer>> location_user_counts = dataStream
.map(new MapFunction<CheckIn, String>() {
    @Override
    public String map(CheckIn value) throws Exception {
        return value.getLocation_id() + "-" + value.getUser();
    }
})
.map(new MapFunction<String, Tuple2<String, Integer>>() {
    @Override
    public Tuple2<String, Integer> map(String value) throws Exception {
        return new Tuple2<String, Integer>(value, 1);
    }
})
.keyBy(new KeySelector<Tuple2<String, Integer>, String>() {
    @Override
    public String getKey(Tuple2<String, Integer> value) throws Exception {
        return value.f0;
    }
})
//.window(SlidingProcessingTimeWindows.of(Time.seconds(10), Time.seconds(10)))
.window(TumblingProcessingTimeWindows.of(Time.seconds(10)))
.sum(1);

location_user_counts.print();

CassandraSink.addSink(location_user_counts)
        .setHost("cassandra")
        .setQuery("INSERT INTO flink.location_user_counts(user_location, counts) values (?, ?);")
        .build();
```

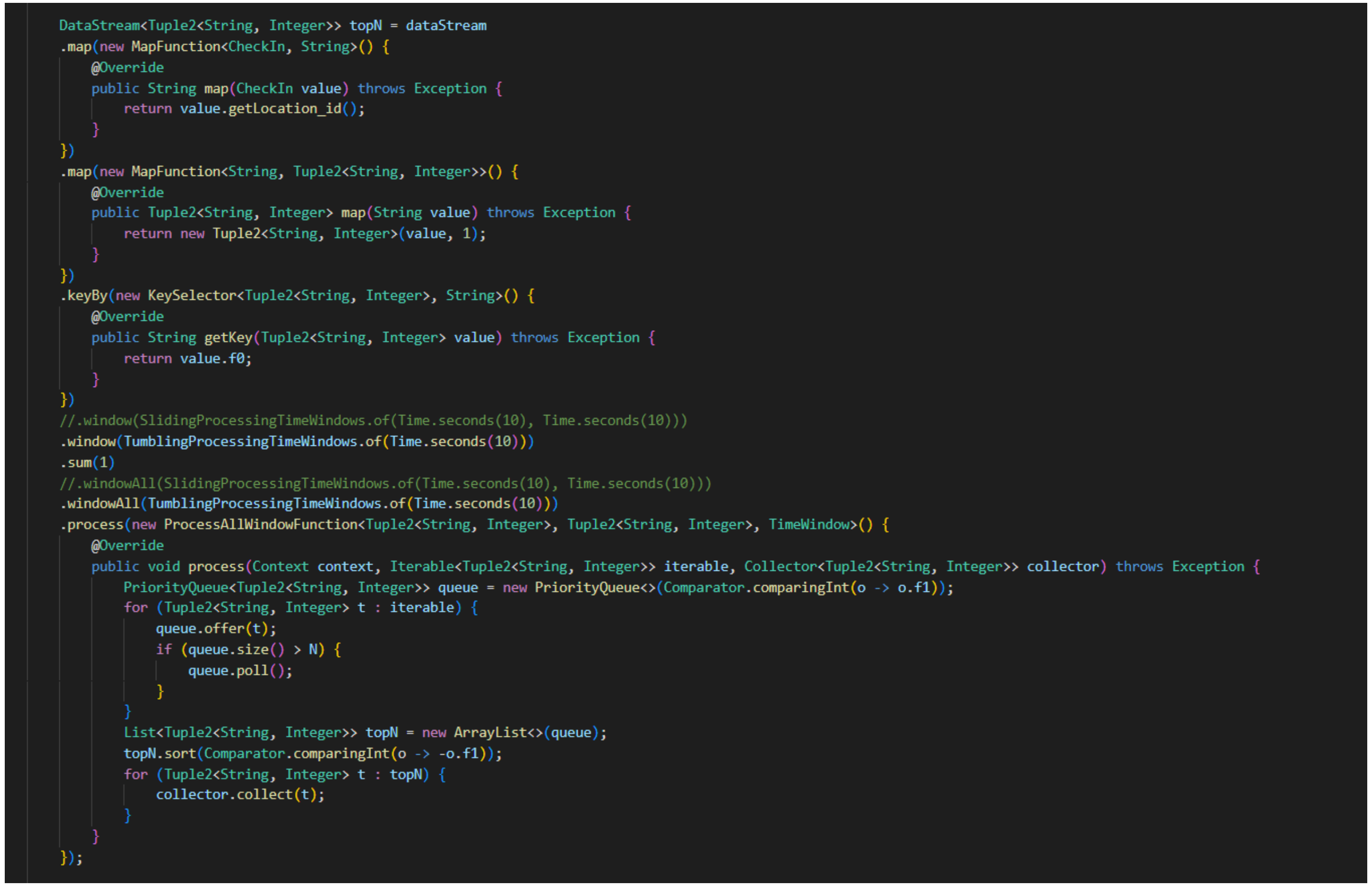
- Određivanje broja korisnika koji je posetio svaku od lokacija i upis rezultata u Cassandra bazu podataka
- Podaci su grupisani u vremenski (po tipu "Tumbling") prozor veličine 10 sekundi

Rezultat u Cassndra DB



```
 user_location                              | counts
--------------------------------------------+--------
         64b925364ac71005af756eabb94096e2-0 |      1
 9848afcc62e500a01cf6fbf24b797732f8963683-0 |      2
         a2451a4a10d0061e87ebcd21b422f44b-0 |      1
         7a0f88982aa015062b95e3b4843f9ca2-0 |      5
         e2460b3f5040fc666cd99128a0c133b7-0 |      1
         974b77e179e872596500b2d22c38bf26-0 |      1
         c83ac485e066dbf68b8619c7ab5b2579-0 |      1
         8f060f74f59a02df0993b24964334eea-0 |      2
         d2f35ed2724211deafe6003048c0801e-0 |      1
         828fb63e770c11ddb8450030487eb504-0 |      1
         115b5206a20011dd9f01003048c10834-0 |      1
         dcc06bf19e775f436c2225be50e14922-0 |      2
         c69bd906c5210caee0719913f836303c-0 |      1
         e63d729e56954aeb23ba669d2c7a2805-0 |      1
         6f5b96170b7744af3c7577fa35ed0b8f-0 |      1
```

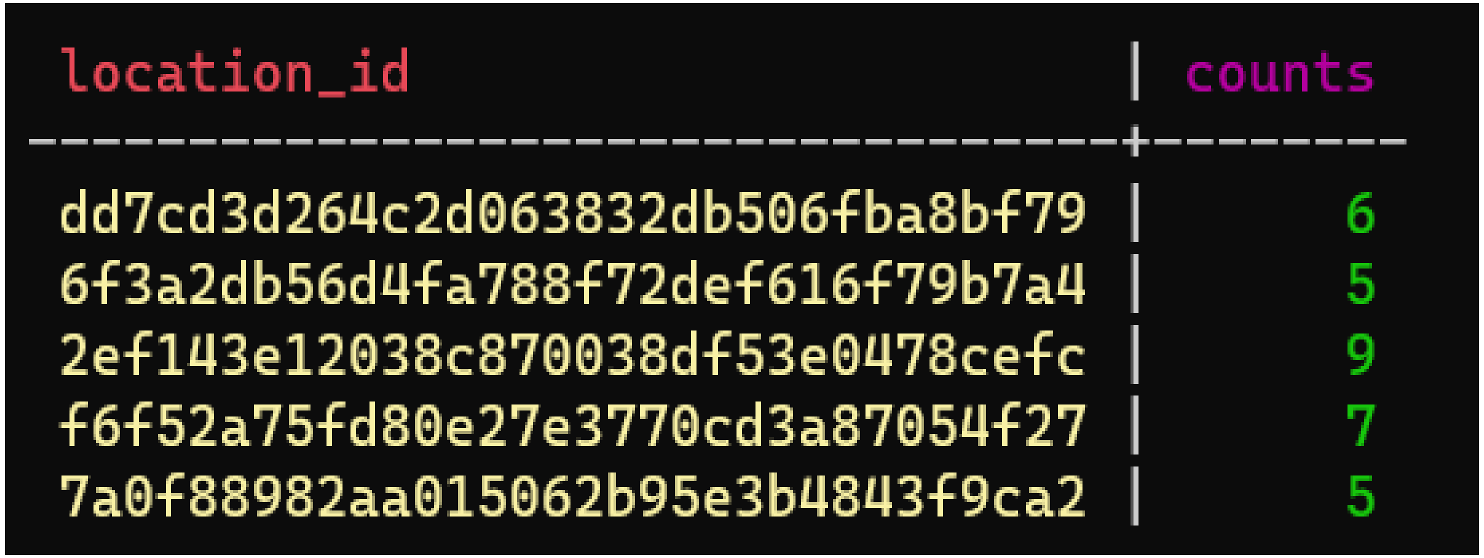
Top N lokacija



```java
DataStream<Tuple2<String, Integer>> topN = dataStream
.map(new MapFunction<CheckIn, String>() {
    @Override
    public String map(CheckIn value) throws Exception {
        return value.getLocation_id();
    }
})
.map(new MapFunction<String, Tuple2<String, Integer>>() {
    @Override
    public Tuple2<String, Integer> map(String value) throws Exception {
        return new Tuple2<String, Integer>(value, 1);
    }
})
.keyBy(new KeySelector<Tuple2<String, Integer>, String>() {
    @Override
    public String getKey(Tuple2<String, Integer> value) throws Exception {
        return value.f0;
    }
})
//.window(SlidingProcessingTimeWindows.of(Time.seconds(10), Time.seconds(10)))
.window(TumblingProcessingTimeWindows.of(Time.seconds(10)))
.sum(1)
//.windowAll(SlidingProcessingTimeWindows.of(Time.seconds(10), Time.seconds(10)))
.windowAll(TumblingProcessingTimeWindows.of(Time.seconds(10)))
.process(new ProcessAllWindowFunction<Tuple2<String, Integer>, Tuple2<String, Integer>, TimeWindow>() {
    @Override
    public void process(Context context, Iterable<Tuple2<String, Integer>> iterable, Collector<Tuple2<String, Integer>> collector) throws Exception {
        PriorityQueue<Tuple2<String, Integer>> queue = new PriorityQueue<>(Comparator.comparingInt(o -> o.f1));
        for (Tuple2<String, Integer> t : iterable) {
            queue.offer(t);
            if (queue.size() > N) {
                queue.poll();
            }
        }
        List<Tuple2<String, Integer>> topN = new ArrayList<>(queue);
        topN.sort(Comparator.comparingInt(o -> -o.f1));
        for (Tuple2<String, Integer> t : topN) {
            collector.collect(t);
        }
    }
});
```

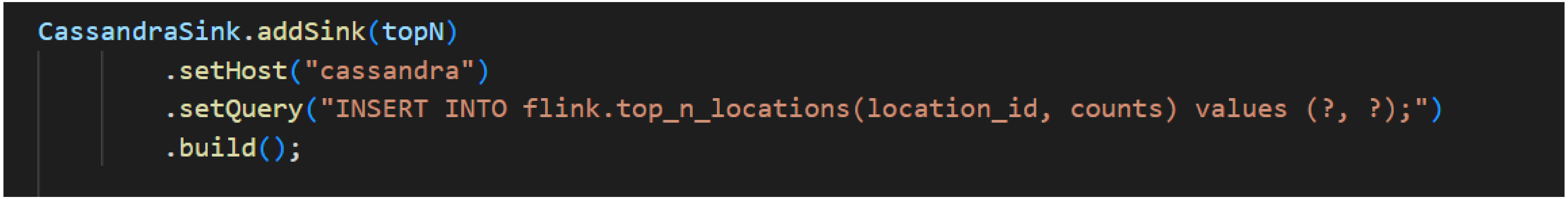
- Pronalazak Top N lokacija u odnosu na broj poseta
- Podaci su grupisani u vremenski (po tipu "Tumbling") prozor veličine 10 sekundi

Rezultat u Cassndra DB



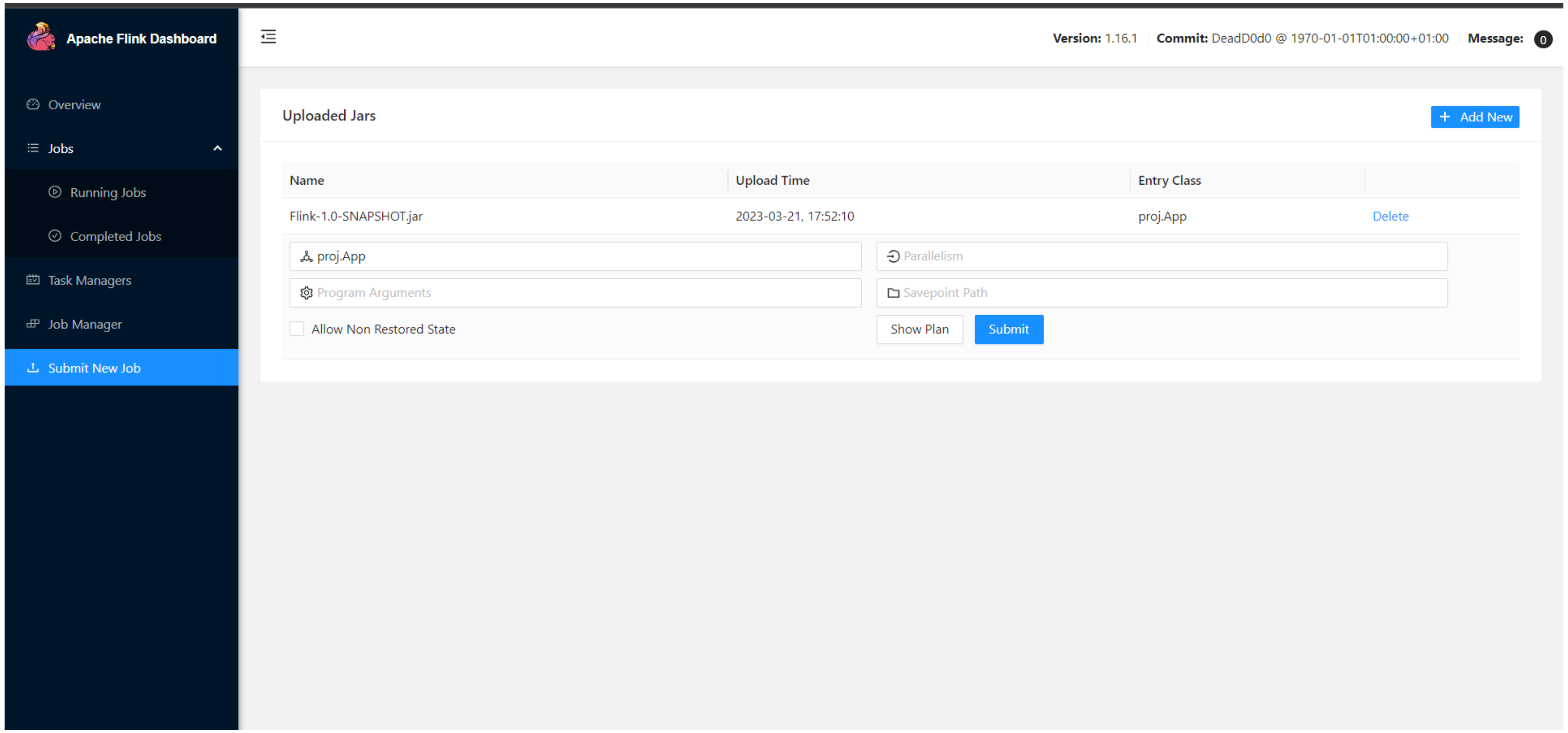
```
 location_id                      | counts
----------------------------------+--------
 dd7cd3d264c2d063832db506fba8bf79 |      6
 6f3a2db56d4fa788f72def616f79b7a4 |      5
 2ef143e12038c870038df53e0478cefc |      9
 f6f52a75fd80e27e3770cd3a87054f27 |      7
 7a0f88982aa015062b95e3b4843f9ca2 |      5
```

Top N lokacija - Cassandra DB



```
CassandraSink.addSink(topN)
        .setHost("cassandra")
        .setQuery("INSERT INTO flink.top_n_locations(location_id, counts) values (?, ?);")
        .build();
```

Web UI



Version: 1.16.1 **Commit:** DeadD0d0 @ 1970-01-01T01:00:00+01:00 **Message:** 0

Uploaded Jars

Name	Upload Time	Entry Class	
Flink-1.0-SNAPSHOT.jar	2023-03-21, 17:52:10	proj.App	Delete

proj.App

Parallelism

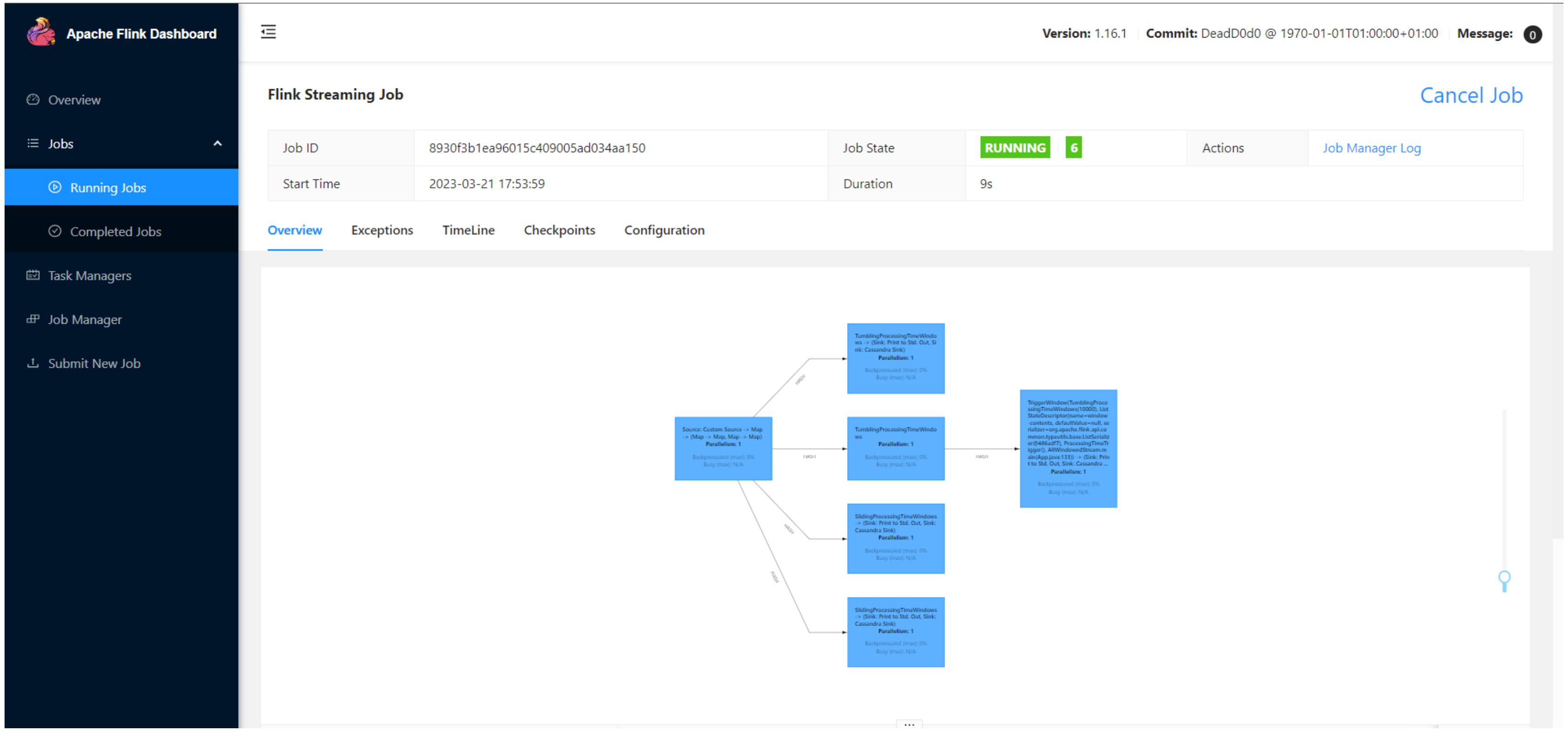
Program Arguments

Savepoint Path

□ Allow Non Restored State

- Aplikacija se startuje tako što se upload-uje željeni JAR fajl preko web UI - a

Web UI



Apache Flink Dashboard
Overview
Jobs
Running Jobs
Completed Jobs
Task Managers
Job Manager
Submit New Job

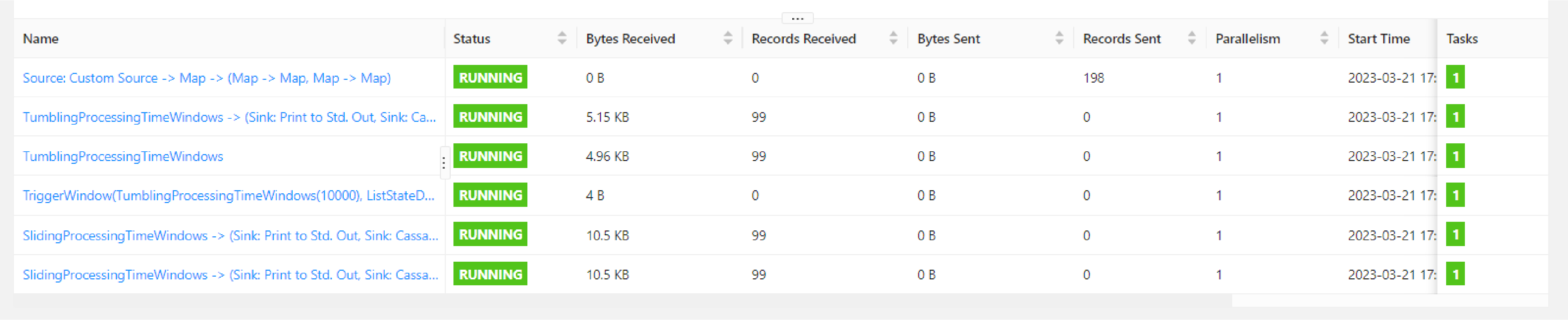
Version: 1.16.1 **Commit:** DeadD0d0 @ 1970-01-01T01:00:00+01:00 **Message:** 0

Flink Streaming Job

Job ID	8930f3b1ea96015c409005ad034aa150	Job State	RUNNING 6	Actions	Job Manager Log
Start Time	2023-03-21 17:53:59	Duration	9s		

Overview
Exceptions
TimeLine
Checkpoints
Configuration
Source: Custom Source -> Map -> (Map -> Map, Map -> Map)
Parallelism: 1
Backpressured (max): 0%
Busy (max): N/A
HASH
TumblingProcessingTimeWindows -> (Sink: Print to Std. Out, Sink: Cassandra Sink)
Parallelism: 1
Backpressured (max): 0%
Busy (max): N/A
HASH
TumblingProcessingTimeWindows
Parallelism: 1
Backpressured (max): 0%
Busy (max): N/A
HASH
TriggerWindow(TumblingProcessingTimeWindows(10000), ListStateDescriptor{name=window-contents, defaultValue=null, serializer=org.apache.flink.api.common.typeutils.base.ListSerializer@486adf7}, ProcessingTimeTrigger(), AllWindowedStream.main(App.java:133)) -> (Sink: Print to Std. Out, Sink: Cassandra ...
Parallelism: 1
Backpressured (max): 0%
Busy (max): N/A
HASH
SlidingProcessingTimeWindows -> (Sink: Print to Std. Out, Sink: Cassandra Sink)
Parallelism: 1
Backpressured (max): 0%
Busy (max): N/A
HASH
SlidingProcessingTimeWindows -> (Sink: Print to Std. Out, Sink: Cassandra Sink)
Parallelism: 1
Backpressured (max): 0%
Busy (max): N/A

Web UI



Name	Status	Bytes Received	Records Received	Bytes Sent	Records Sent	Parallelism	Start Time	Tasks
Source: Custom Source -> Map -> (Map -> Map, Map -> Map)	RUNNING	0 B	0	0 B	198	1	2023-03-21 17:	1
TumblingProcessingTimeWindows -> (Sink: Print to Std. Out, Sink: Ca...	RUNNING	5.15 KB	99	0 B	0	1	2023-03-21 17:	1
TumblingProcessingTimeWindows	RUNNING	4.96 KB	99	0 B	0	1	2023-03-21 17:	1
TriggerWindow(TumblingProcessingTimeWindows(10000), ListStateD...	RUNNING	4 B	0	0 B	0	1	2023-03-21 17:	1
SlidingProcessingTimeWindows -> (Sink: Print to Std. Out, Sink: Cassa...	RUNNING	10.5 KB	99	0 B	0	1	2023-03-21 17:	1
SlidingProcessingTimeWindows -> (Sink: Print to Std. Out, Sink: Cassa...	RUNNING	10.5 KB	99	0 B	0	1	2023-03-21 17:	1