Miloš Veljanovski 1559

Projekat 2 – Sistemi za analizu velike količine podataka

Korišćeni izvorni podaci

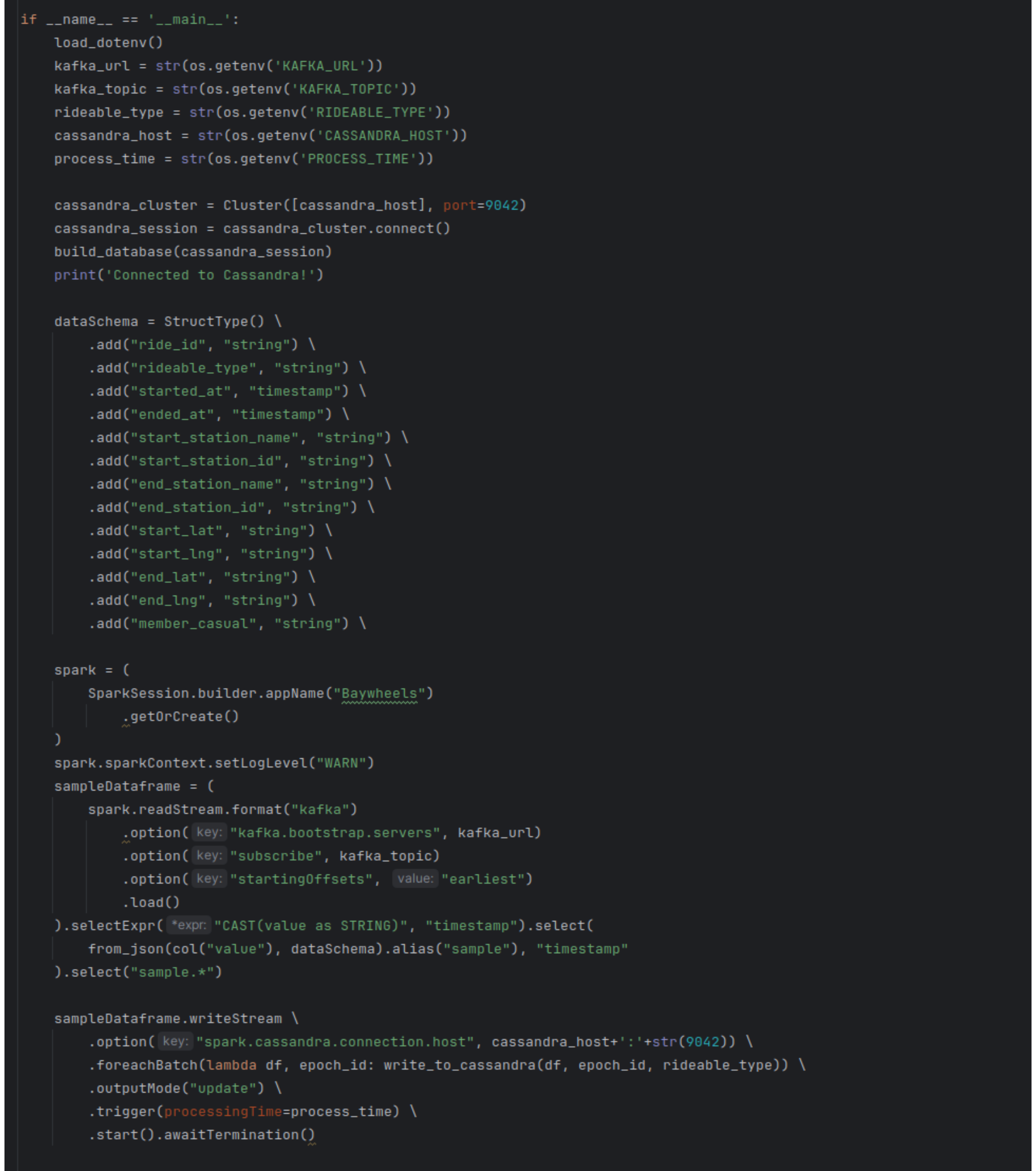
- Iskorišćen je set podataka o vožnji biciklama iz „oblasti zaliva“ tj. San Francisko zaliva (uključujući Palo Alto i San Hoze)
- Dataset sadrži sledeće kolene podataka:
  - Vreme i datum početka i završetka vožnje
  - Podaci o početkoj stanici (ID, ime, geografska širina i dužina)
  - Podaci o završnoj stanici (ID, ime, geografska širina i dužina)
  - ID vožnje
  - Tip korisnika (vozača)
- Korišćeni su podaci za 2021. i 2022. godinu

PySpark stream obrada podataka

- Kafka Producer i Consumer napisani su u Python-u
- Streaming obrada podataka vrši se nad vrstama dataset-a poslatih na Kafka topic, u vremenskim prozorima od 2 sekunde
- U okviru aplikacije, realizovana su 2 zadatka (task-a)
  - Za prvi zadatak, posmatraju se vožnje sa električnom biciklom i izračunavaju se minimalno, maksimalno i prosečno trajanje vožnje
  - Za drugi zadatak, posmatraju se sve vožnje i pronalaze tri najpopularnije stanice prema broju završenih vožnji
- Upis rezultata vrši se u Cassandra bazu podataka

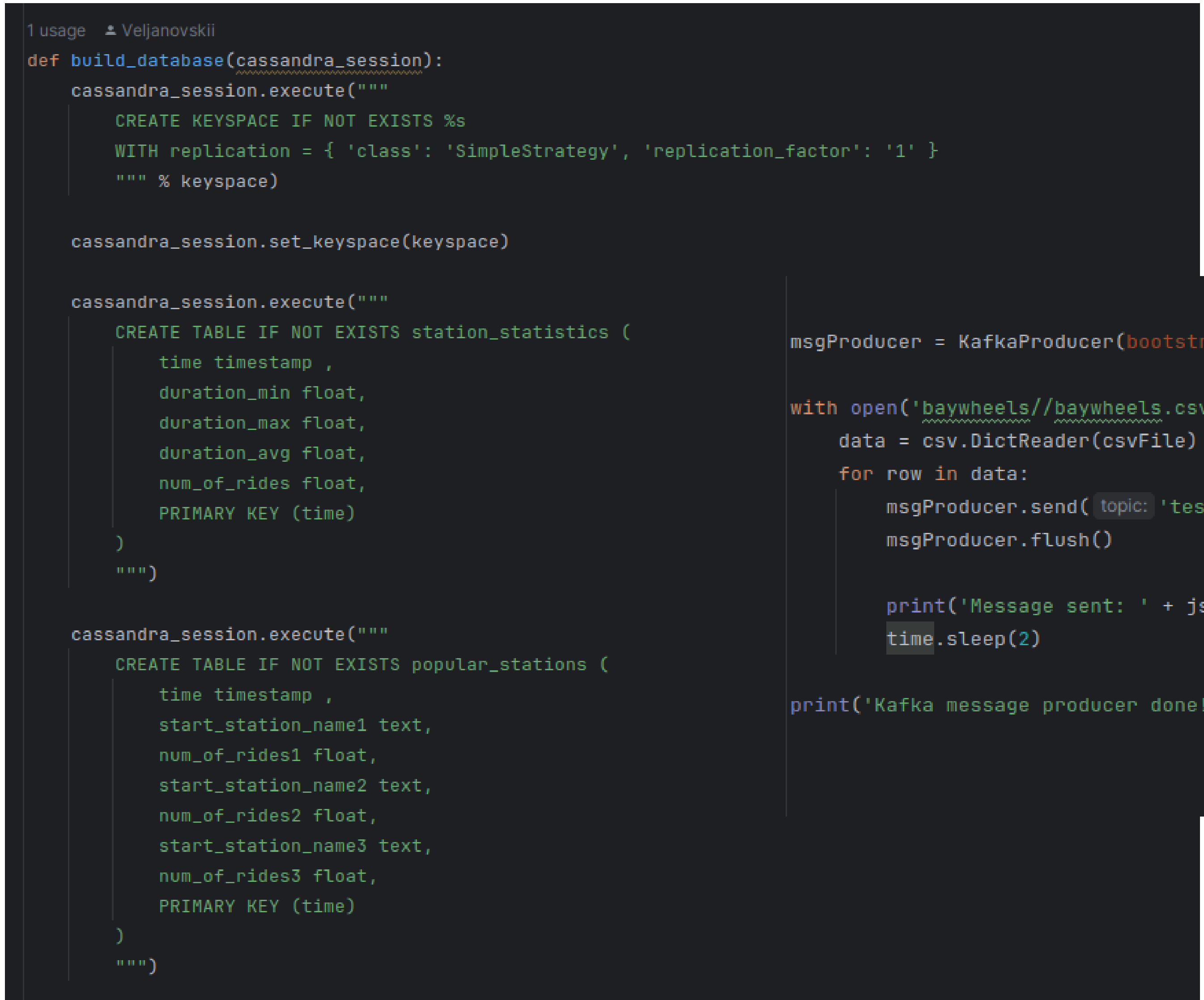
Delovi koda Spark aplikacije

main.py



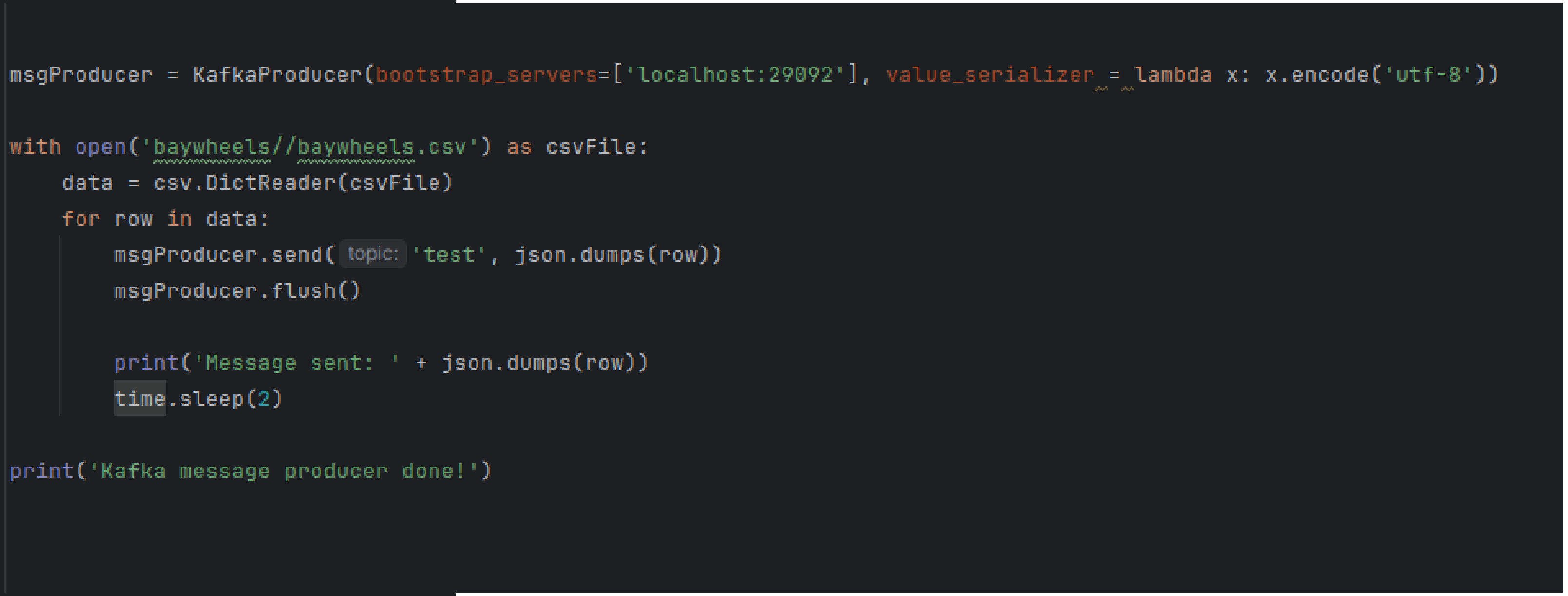
```python
if __name__ == '__main__':
    load_dotenv()
    kafka_url = str(os.getenv('KAFKA_URL'))
    kafka_topic = str(os.getenv('KAFKA_TOPIC'))
    rideable_type = str(os.getenv('RIDEABLE_TYPE'))
    cassandra_host = str(os.getenv('CASSANDRA_HOST'))
    process_time = str(os.getenv('PROCESS_TIME'))

    cassandra_cluster = Cluster([cassandra_host], port=9042)
    cassandra_session = cassandra_cluster.connect()
    build_database(cassandra_session)
    print('Connected to Cassandra!')

    dataSchema = StructType() \
        .add("ride_id", "string") \
        .add("rideable_type", "string") \
        .add("started_at", "timestamp") \
        .add("ended_at", "timestamp") \
        .add("start_station_name", "string") \
        .add("start_station_id", "string") \
        .add("end_station_name", "string") \
        .add("end_station_id", "string") \
        .add("start_lat", "string") \
        .add("start_lng", "string") \
        .add("end_lat", "string") \
        .add("end_lng", "string") \
        .add("member_casual", "string") \

    spark = (
        SparkSession.builder.appName("Baywheels")
            .getOrCreate()
    )
    spark.sparkContext.setLogLevel("WARN")
    sampleDataframe = (
        spark.readStream.format("kafka")
            .option( key: "kafka.bootstrap.servers", kafka_url)
            .option( key: "subscribe", kafka_topic)
            .option( key: "startingOffsets",  value: "earliest")
            .load()
    ).selectExpr( *expr: "CAST(value as STRING)", "timestamp").select(
        from_json(col("value"), dataSchema).alias("sample"), "timestamp"
    ).select("sample.*")

    sampleDataframe.writeStream \
        .option( key: "spark.cassandra.connection.host", cassandra_host+':'+str(9042)) \
        .foreachBatch(lambda df, epoch_id: write_to_cassandra(df, epoch_id, rideable_type)) \
        .outputMode("update") \
        .trigger(processingTime=process_time) \
        .start().awaitTermination()
```

Delovi koda Spark aplikacije



```
1 usage  Veljanovskii
def build_database(cassandra_session):
    cassandra_session.execute("""
        CREATE KEYSPACE IF NOT EXISTS %s
        WITH replication = { 'class': 'SimpleStrategy', 'replication_factor': '1' }
        """ % keyspace)

    cassandra_session.set_keyspace(keyspace)

    cassandra_session.execute("""
        CREATE TABLE IF NOT EXISTS station_statistics (
            time timestamp ,
            duration_min float,
            duration_max float,
            duration_avg float,
            num_of_rides float,
            PRIMARY KEY (time)
        )
        """)

    cassandra_session.execute("""
        CREATE TABLE IF NOT EXISTS popular_stations (
            time timestamp ,
            start_station_name1 text,
            num_of_rides1 float,
            start_station_name2 text,
            num_of_rides2 float,
            start_station_name3 text,
            num_of_rides3 float,
            PRIMARY KEY (time)
        )
        """)
```

Upis podataka u bazu

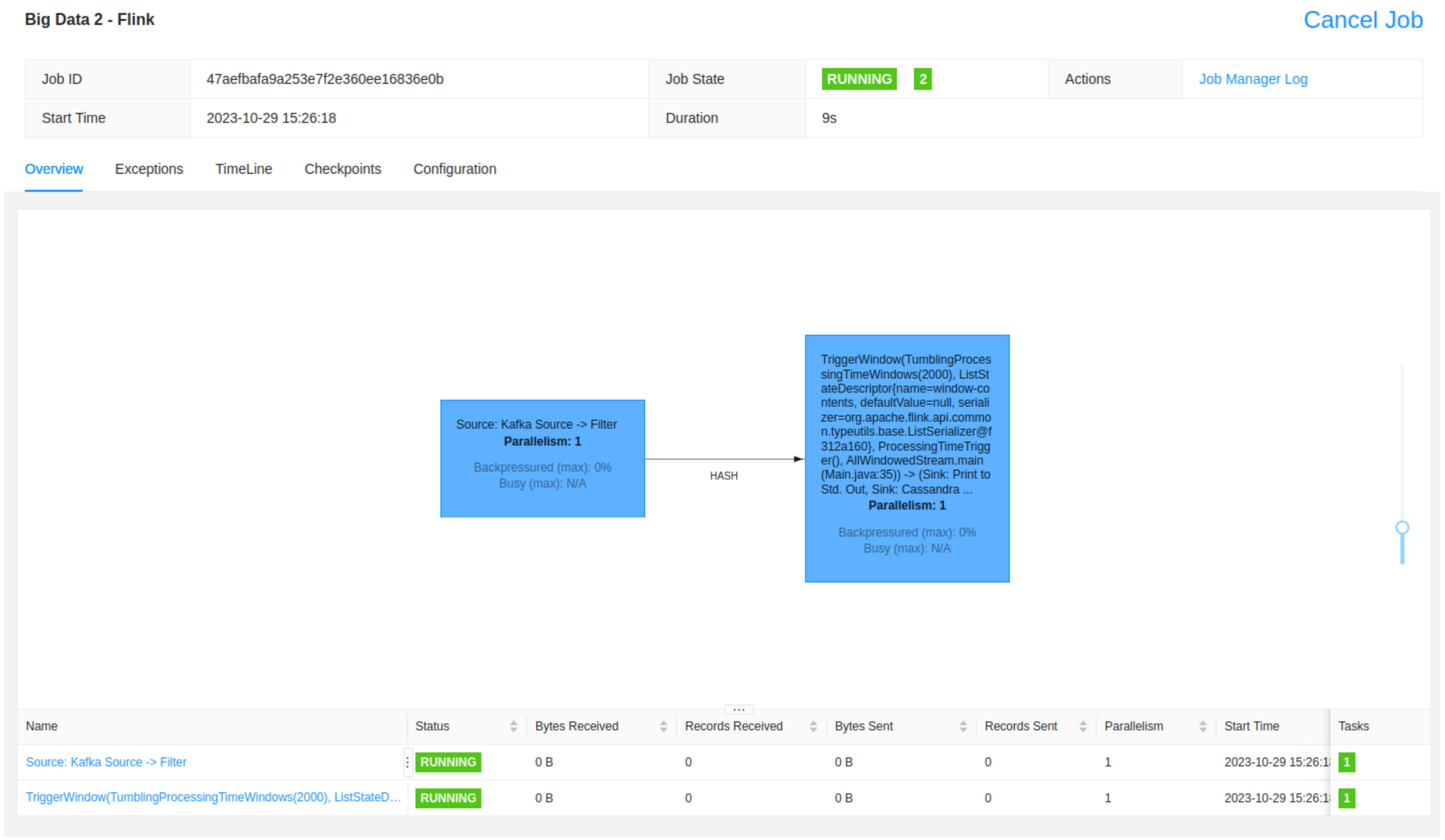


```
msgProducer = KafkaProducer(bootstrap_servers=['localhost:29092'], value_serializer = lambda x: x.encode('utf-8'))

with open('baywheels//baywheels.csv') as csvFile:
    data = csv.DictReader(csvFile)
    for row in data:
        msgProducer.send( topic: 'test', json.dumps(row))
        msgProducer.flush()

        print('Message sent: ' + json.dumps(row))
        time.sleep(2)

print('Kafka message producer done!')
```

Kafka producer

Flink stream obrada podataka

Izvršenje na Flink klasteru



Big Data 2 - Flink

Cancel Job

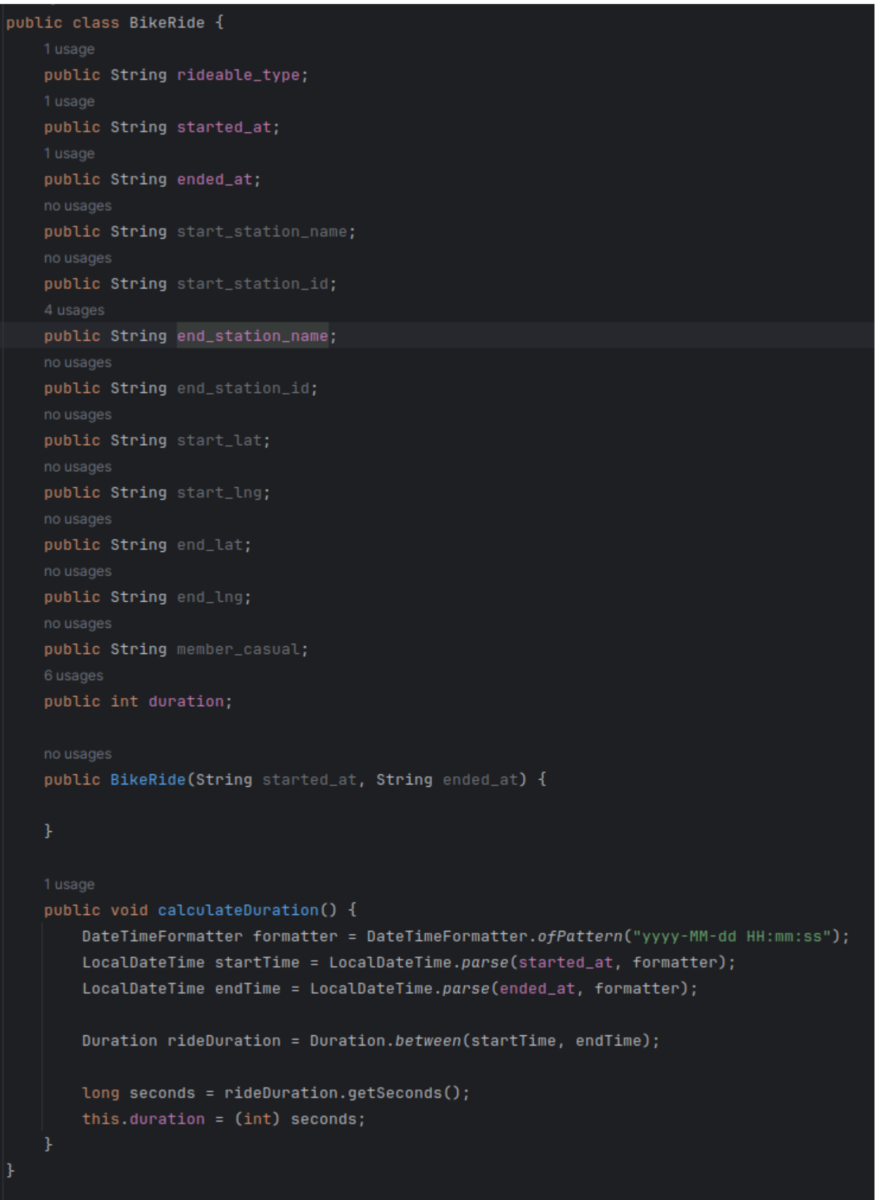
Job ID	47aefbafa9a253e7f2e360ee16836e0b	Job State	RUNNING 2	Actions	Job Manager Log
Start Time	2023-10-29 15:26:18	Duration	9s		

Overview Exceptions TimeLine Checkpoints Configuration

Source: Kafka Source -> Filter
Parallelism: 1
Backpressured (max): 0%
Busy (max): N/A
HASH
TriggerWindow(TumblingProcessingTimeWindows(2000), ListStateDescriptor{name=window-contents, defaultValue=null, serializer=org.apache.flink.api.common.typeutils.base.ListSerializer@f312a160}, ProcessingTimeTrigger(), AllWindowedStream.main(Main.java:35)) -> (Sink: Print to Std. Out, Sink: Cassandra ...
Parallelism: 1
Backpressured (max): 0%
Busy (max): N/A

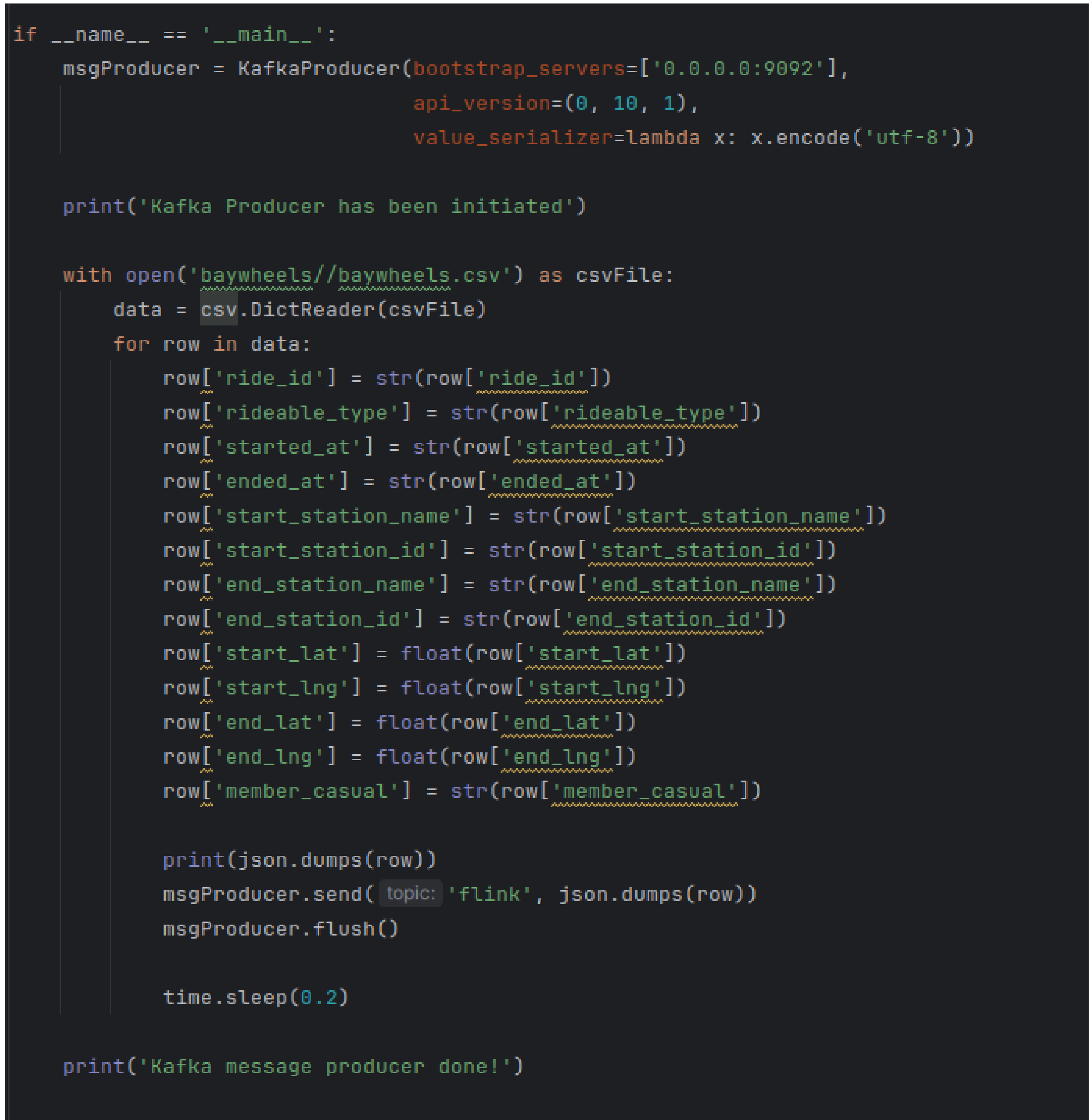
Name	Status	Bytes Received	Records Received	Bytes Sent	Records Sent	Parallelism	Start Time	Tasks
Source: Kafka Source -> Filter	RUNNING	0 B	0	0 B	0	1	2023-10-29 15:26:1	1
TriggerWindow(TumblingProcessingTimeWindows(2000), ListStateD…	RUNNING	0 B	0	0 B	0	1	2023-10-29 15:26:1	1

Delovi koda Flink aplikacije



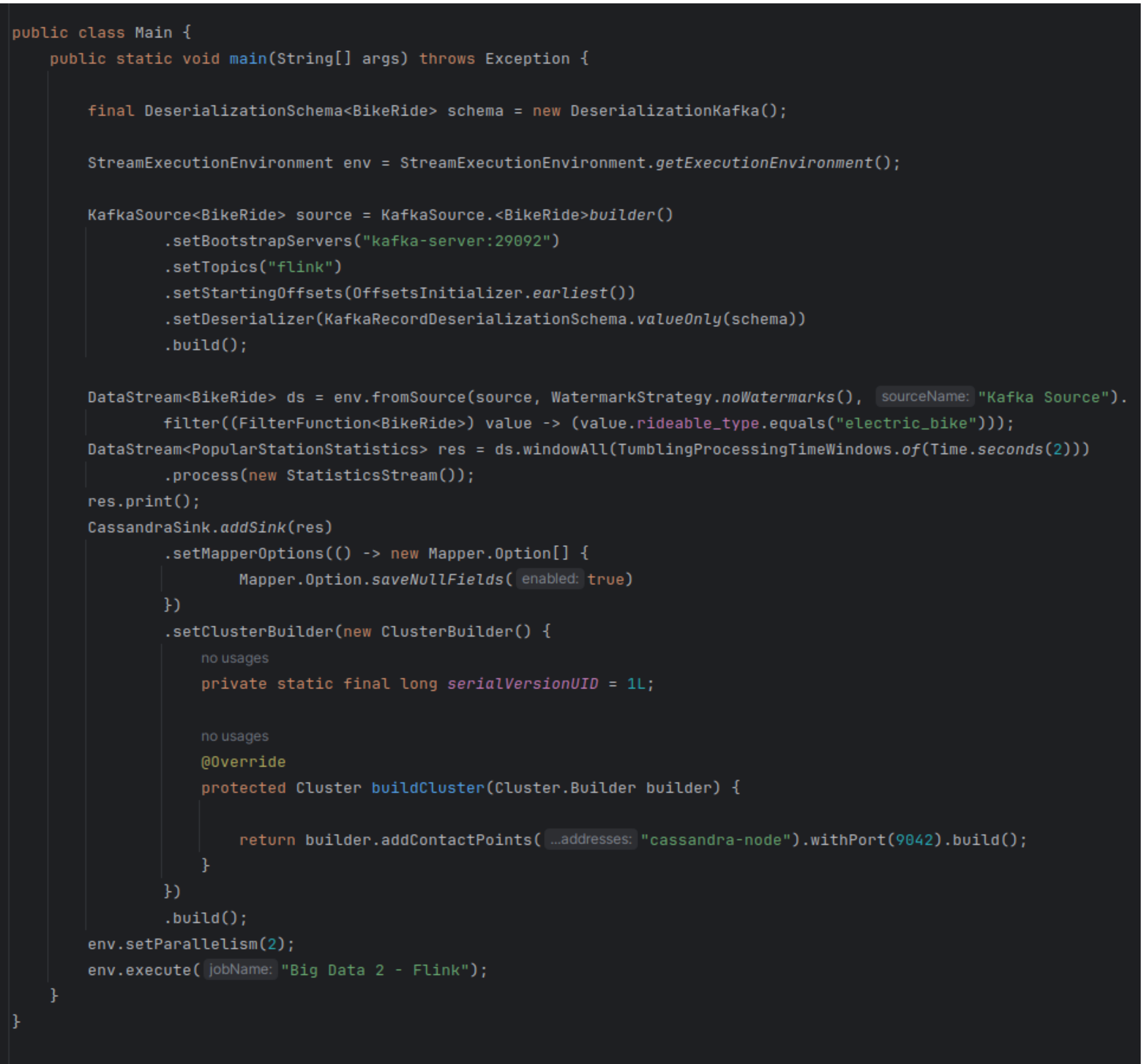
```java
public class BikeRide {
    1 usage
    public String rideable_type;
    1 usage
    public String started_at;
    1 usage
    public String ended_at;
    no usages
    public String start_station_name;
    no usages
    public String start_station_id;
    4 usages
    public String end_station_name;
    no usages
    public String end_station_id;
    no usages
    public String start_lat;
    no usages
    public String start_lng;
    no usages
    public String end_lat;
    no usages
    public String end_lng;
    no usages
    public String member_casual;
    6 usages
    public int duration;

    no usages
    public BikeRide(String started_at, String ended_at) {

    }

    1 usage
    public void calculateDuration() {
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss");
        LocalDateTime startTime = LocalDateTime.parse(started_at, formatter);
        LocalDateTime endTime = LocalDateTime.parse(ended_at, formatter);

        Duration rideDuration = Duration.between(startTime, endTime);

        long seconds = rideDuration.getSeconds();
        this.duration = (int) seconds;
    }
}
```

Flink producer

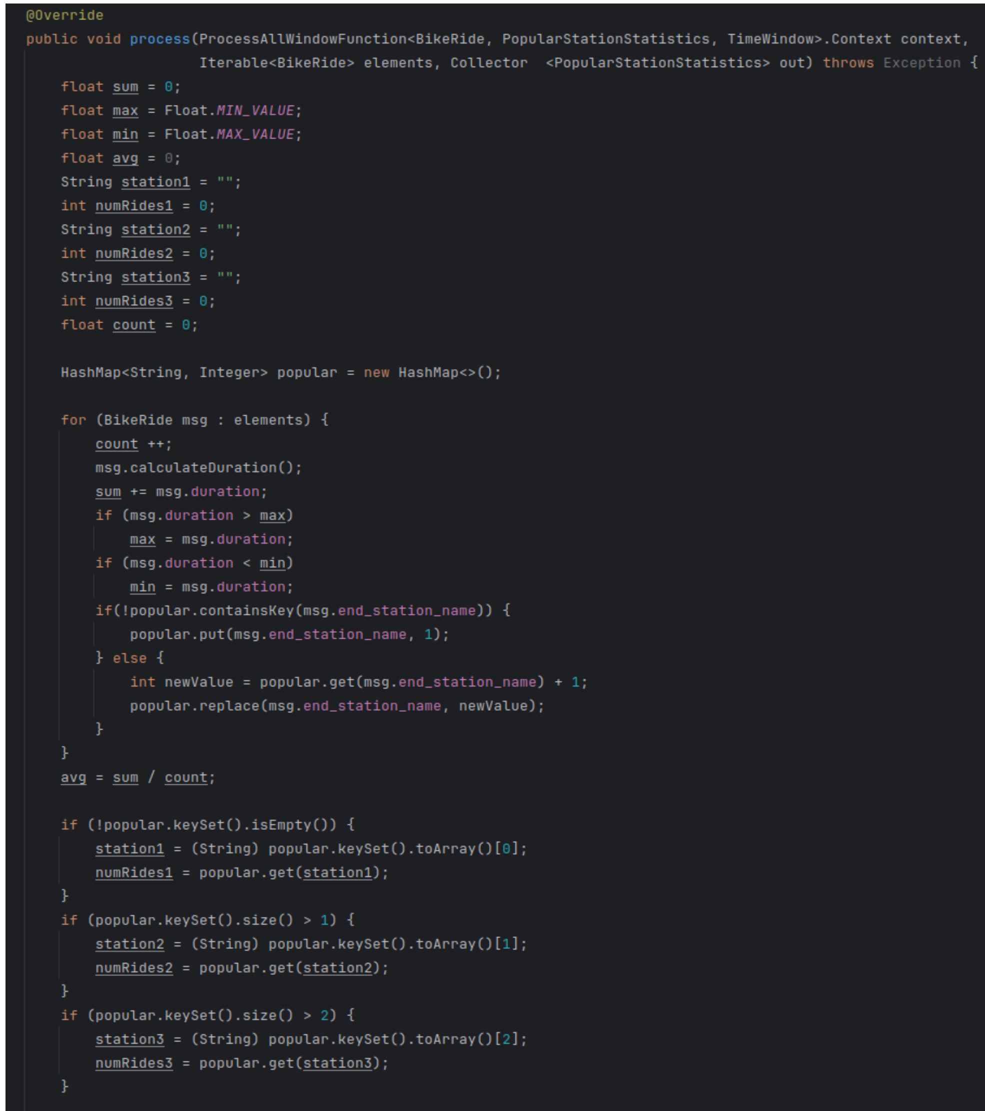


```python
if __name__ == '__main__':
    msgProducer = KafkaProducer(bootstrap_servers=['0.0.0.0:9092'],
                                api_version=(0, 10, 1),
                                value_serializer=lambda x: x.encode('utf-8'))

    print('Kafka Producer has been initiated')

    with open('baywheels//baywheels.csv') as csvFile:
        data = csv.DictReader(csvFile)
        for row in data:
            row['ride_id'] = str(row['ride_id'])
            row['rideable_type'] = str(row['rideable_type'])
            row['started_at'] = str(row['started_at'])
            row['ended_at'] = str(row['ended_at'])
            row['start_station_name'] = str(row['start_station_name'])
            row['start_station_id'] = str(row['start_station_id'])
            row['end_station_name'] = str(row['end_station_name'])
            row['end_station_id'] = str(row['end_station_id'])
            row['start_lat'] = float(row['start_lat'])
            row['start_lng'] = float(row['start_lng'])
            row['end_lat'] = float(row['end_lat'])
            row['end_lng'] = float(row['end_lng'])
            row['member_casual'] = str(row['member_casual'])

            print(json.dumps(row))
            msgProducer.send( topic: 'flink', json.dumps(row))
            msgProducer.flush()

            time.sleep(0.2)

    print('Kafka message producer done!')
```

Delovi koda Flink aplikacije



```java
public class Main {
    public static void main(String[] args) throws Exception {

        final DeserializationSchema<BikeRide> schema = new DeserializationKafka();

        StreamExecutionEnvironment env = StreamExecutionEnvironment.getExecutionEnvironment();

        KafkaSource<BikeRide> source = KafkaSource.<BikeRide>builder()
                .setBootstrapServers("kafka-server:29092")
                .setTopics("flink")
                .setStartingOffsets(OffsetsInitializer.earliest())
                .setDeserializer(KafkaRecordDeserializationSchema.valueOnly(schema))
                .build();

        DataStream<BikeRide> ds = env.fromSource(source, WatermarkStrategy.noWatermarks(), sourceName: "Kafka Source").
                filter((FilterFunction<BikeRide>) value -> (value.rideable_type.equals("electric_bike")));
        DataStream<PopularStationStatistics> res = ds.windowAll(TumblingProcessingTimeWindows.of(Time.seconds(2)))
                .process(new StatisticsStream());
        res.print();
        CassandraSink.addSink(res)
                .setMapperOptions(() -> new Mapper.Option[] {
                        Mapper.Option.saveNullFields( enabled: true)
                })
                .setClusterBuilder(new ClusterBuilder() {
                    no usages
                    private static final long serialVersionUID = 1L;

                    no usages
                    @Override
                    protected Cluster buildCluster(Cluster.Builder builder) {

                        return builder.addContactPoints( ...addresses: "cassandra-node").withPort(9042).build();
                    }
                })
                .build();
        env.setParallelism(2);
        env.execute( jobName: "Big Data 2 - Flink");
    }
}
```



```java
@Override
public void process(ProcessAllWindowFunction<BikeRide, PopularStationStatistics, TimeWindow>.Context context,
                    Iterable<BikeRide> elements, Collector  <PopularStationStatistics> out) throws Exception {
    float sum = 0;
    float max = Float.MIN_VALUE;
    float min = Float.MAX_VALUE;
    float avg = 0;
    String station1 = "";
    int numRides1 = 0;
    String station2 = "";
    int numRides2 = 0;
    String station3 = "";
    int numRides3 = 0;
    float count = 0;

    HashMap<String, Integer> popular = new HashMap<>();

    for (BikeRide msg : elements) {
        count ++;
        msg.calculateDuration();
        sum += msg.duration;
        if (msg.duration > max)
            max = msg.duration;
        if (msg.duration < min)
            min = msg.duration;
        if(!popular.containsKey(msg.end_station_name)) {
            popular.put(msg.end_station_name, 1);
        } else {
            int newValue = popular.get(msg.end_station_name) + 1;
            popular.replace(msg.end_station_name, newValue);
        }
    }
    avg = sum / count;

    if (!popular.keySet().isEmpty()) {
        station1 = (String) popular.keySet().toArray()[0];
        numRides1 = popular.get(station1);
    }
    if (popular.keySet().size() > 1) {
        station2 = (String) popular.keySet().toArray()[1];
        numRides2 = popular.get(station2);
    }
    if (popular.keySet().size() > 2) {
        station3 = (String) popular.keySet().toArray()[2];
        numRides3 = popular.get(station3);
    }
```