ALTIMETRIK®

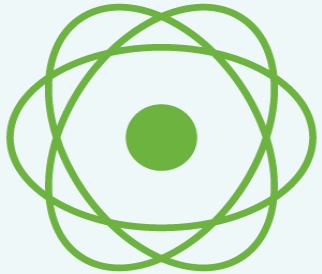
# Reaktywna zabawa przy filiżance Javy

# Agenda

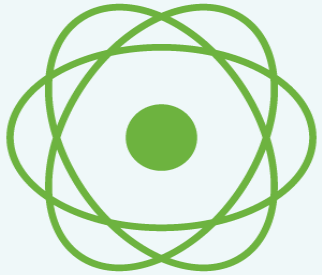
1. Reactive Streams
2. Struktura projektu
3. Plik -> Flux
4. Flux -> KSQL
5. KSQL -> Mutiny

# Reactive Streams

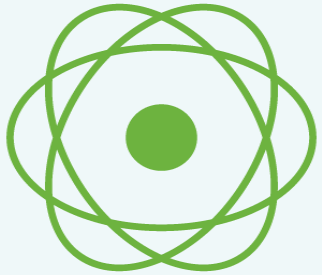
# Biblioteki



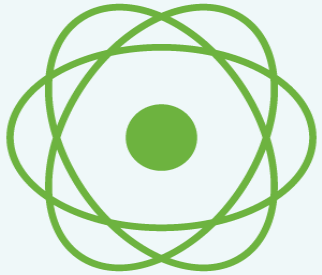
# Biblioteki



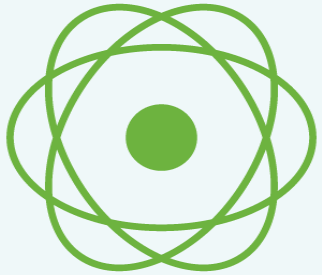
# Biblioteki



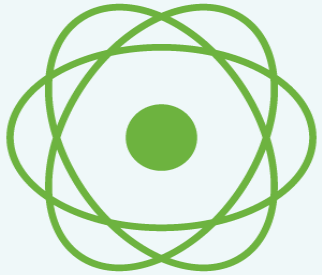
# Biblioteki



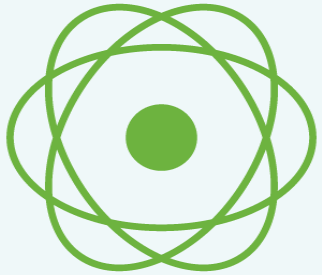

# Biblioteki

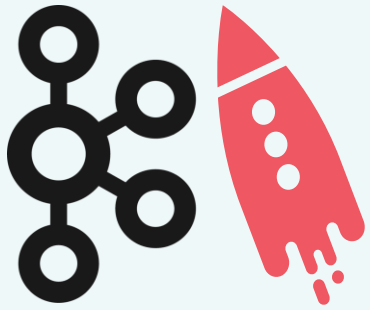


# Biblioteki

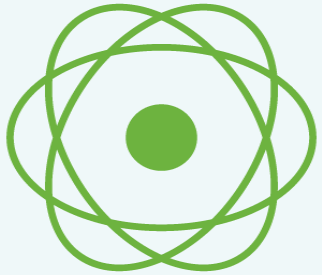


# Biblioteki

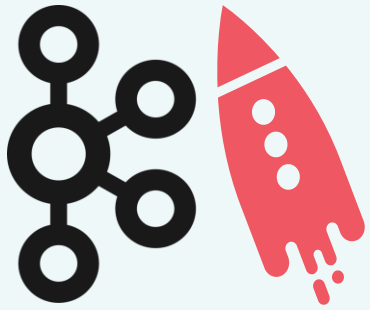




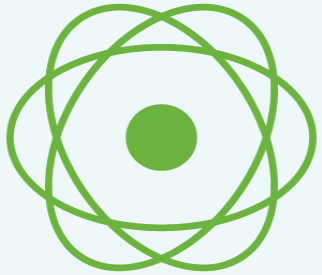
# Biblioteki



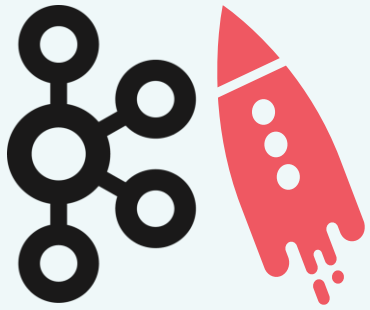


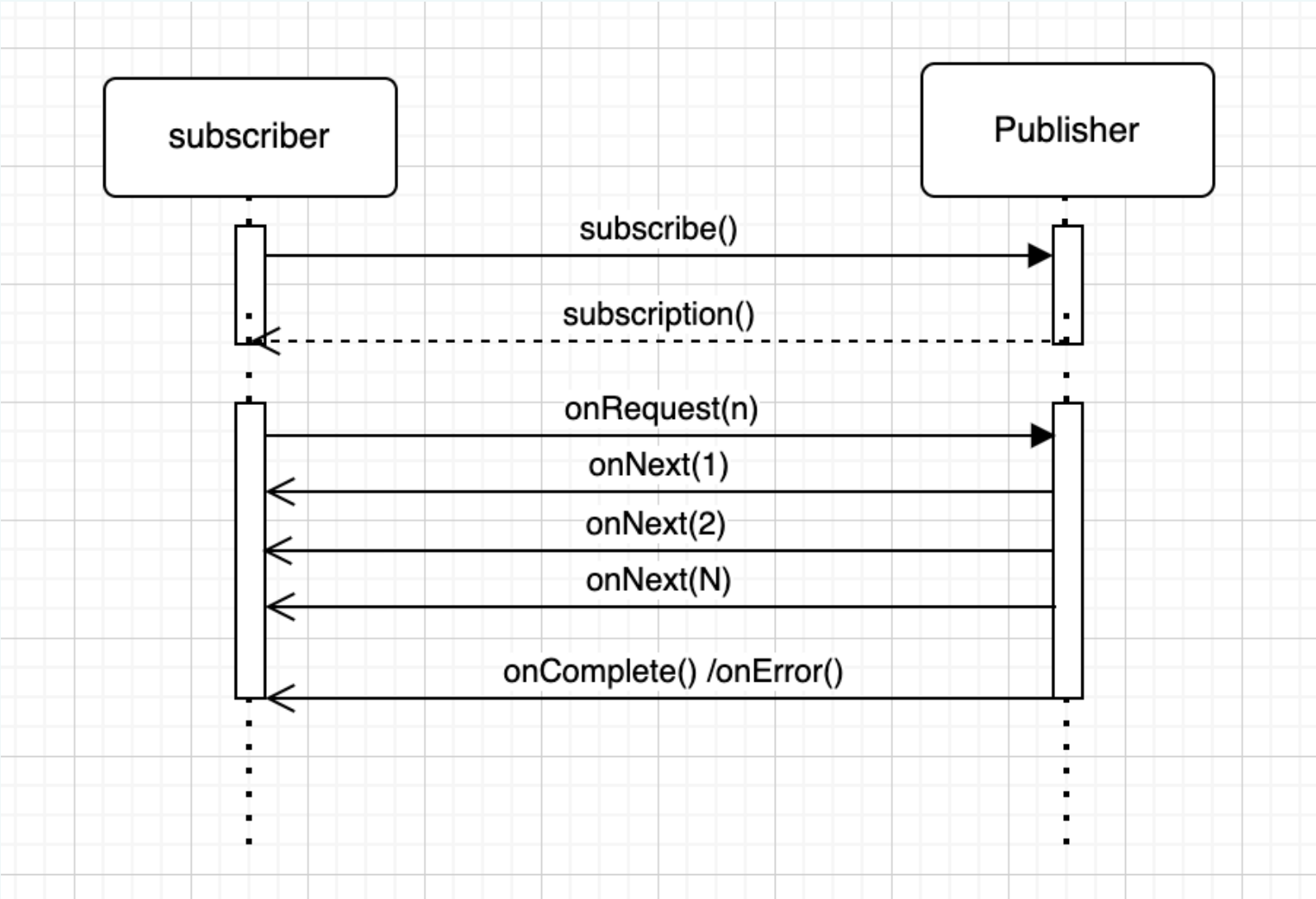

# Biblioteki





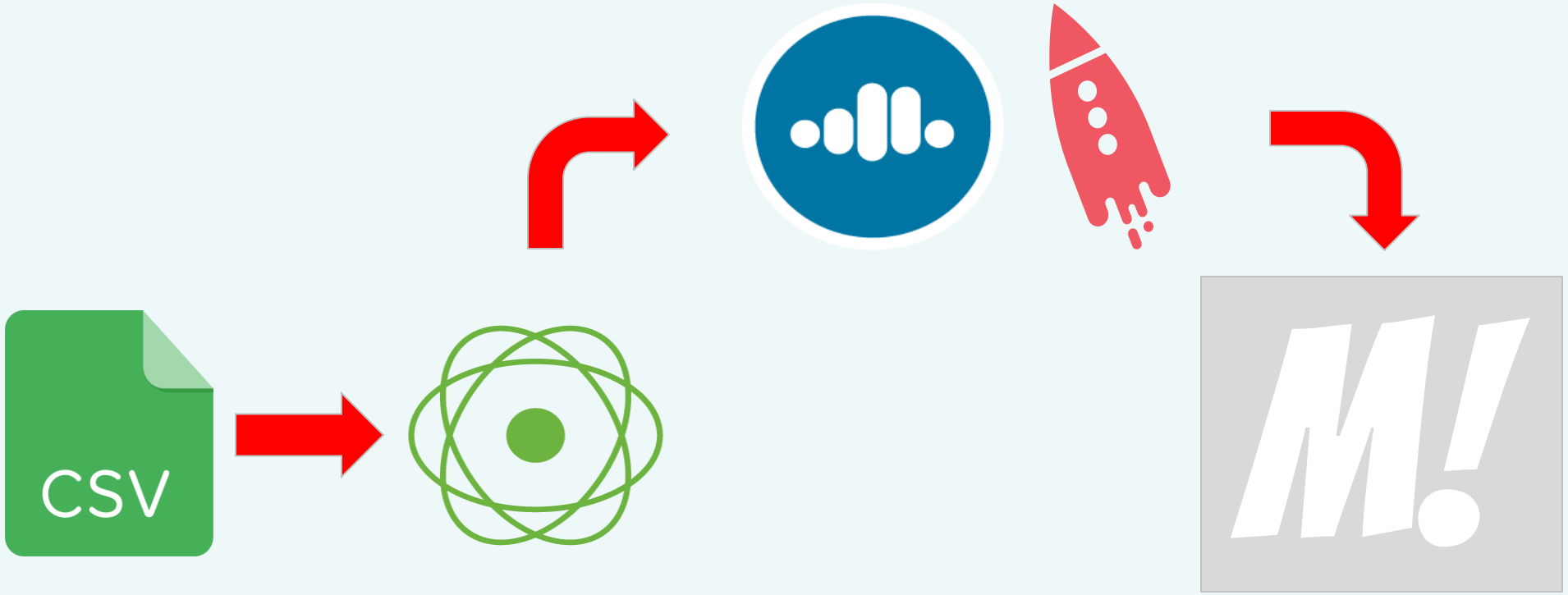



# Sygnały


subscriber
Publisher
subscribe()
subscription()
onRequest(n)
onNext(1)
onNext(2)
onNext(N)
onComplete() /onError()

# Struktura projektu

# Floł



# Plik -> Flux

# Kod

```
return Flux.using(
        () -> new CSVReader(reader),
        Flux::fromIterable);
```

# Flux -> KSQL

# Kod

```
return Mono.fromFuture(client.streamInserts("SAMPLE_STREAM", insertsPublisher));
```

# Kod

```
return Mono.fromFuture(client.streamInserts("SAMPLE_STREAM", insertsPublisher));
```

# KSQL -> Mutiny

# Kod

```
Uni.createFrom().future(client.streamQuery(sql, PROPERTIES))
        .onItem().transformToMulti(JdkFlowAdapter::publisherToFlowPublisher)
```

# Kod

```
Uni.createFrom().future(client.streamQuery(sql, PROPERTIES))
        .onItem().transformToMulti(JdkFlowAdapter::publisherToFlowPublisher)
```

# Cytat

**Attention**

Reactor still uses the legacy Reactive Streams APIs instead of `java.util.concurrent.Flow`, so you need to perform an adaptation.

We recommend using the Mutiny Zero Flow Adapters library as in these examples (Maven coordinates `io.smallrye.reactive:mutiny-zero-flow-adapters`).

ALTIMETRIK®

# Dzięki

**Przydatne linki:**

- reactive-streams.org
- reactivemanifesto.org
- smallrye.io/smallrye-mutiny/latest
- ksqldb.io
- medium.com - porównanie bibliotek