RTP Documentation

Why I choose the path I choose ?

Who knows…

1) Technologies:

   I choose the Erlang Programming language because advantages of fast way of compiling and designed work with actors which are essential in this laboratory work

   a) **Erlang vs Elixir**: Elixir is created on top of the Erlang and in any case for good knowledge of Elixir the developer must know also good Erlang which is double work
   b) **Erlang vs Scala:** Scala is based upon JVM and OOP approach that means that in terms of immutability we can occur data changes or loss during runtime that can cause a lot of debugging, also hot swapping is partially implemented giving opportunity to change only the content of methods but not methods themselves

Actors:

1. **'Application'** - actor to start and stop 'Main Supervisor'.
2. **'Main Supervisor'** - actor to start 'SSE listener Supervisor', 'Worker Supervisor', 'Worker Manager' and 'Worker Scaler'.
3. **'SSE listener Supervisor'** - actor by initialization get a list of URLs and for every URL start a new 'SSE listener' actor.
4. **'SSE listener'** - actor to establish a SEE connection, listen to all events and send json data to 'Worker Manager'.
5. **'Worker Manager'** - actor to receive json data from listener, notify 'Worker Scaler' about new message, choose 'Worker' by 'round-robin distribution' and send json data to this 'Worker'.
6. **'Worker Scaler'** - actor to start new 'Workers' by 'Worker Supervisor' or stop useless. Actor will count messages in a time interval and decide how many 'Workers' are enough.
7. **'Worker Supervisor'** - actor to dynamically balance the amount of 'Workers'.
8. **'Worker'** - actor to receive json data from 'Worker Manager', serialize it, sleep for a half of a second to imitate some processing and print data in output.

Endpoints:

1. **‘Worker Manager’**:

   i. Async: {tweet - atom, Tweet - json data string}
   Get json data to send it to one of ‘Workers’.

2. **‘Worker Scaler’**:

   i. Async: {inc - atom}
   Get signal to increment current counter.

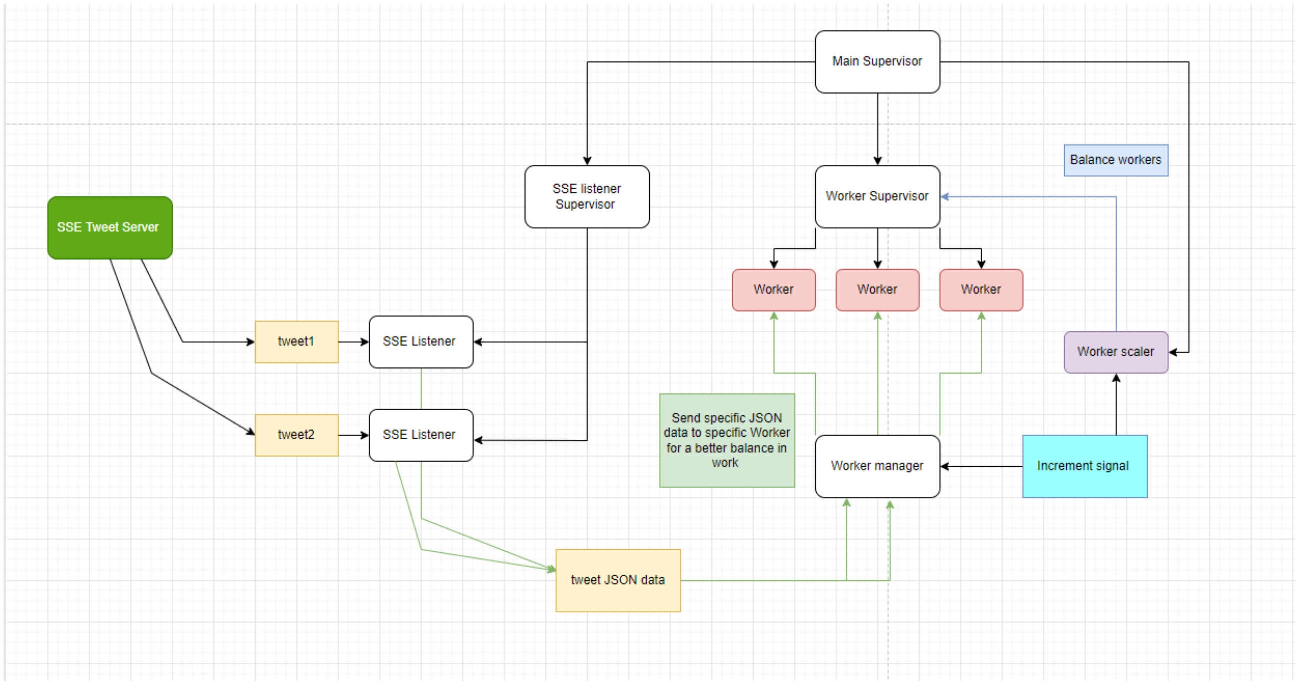
3. **‘Worker’**:

   i. Async: {tweet - atom, Tweet - json data string}
   Process tweet.

4. **‘Worker Supervisor’**:

   i. Function call: **start_worker/1**, when Count - integer
   Start ‘Count’ amount of workers.

   ii. Function call: **stop_worker/1**, when Count - integer
   Stop ‘Count’ amount of workers.


Main Supervisor
SSE listener Supervisor
Worker Supervisor
Balance workers
SSE Tweet Server
Worker
Worker
Worker
tweet1
SSE Listener
Worker scaler
tweet2
SSE Listener
Send specific JSON data to specific Worker for a better balance in work
Worker manager
Increment signal
tweet JSON data