RTP Documentation.

1. **Technologies**:
   a. **Erlang language** - I chose this language from the list of available ones [Erlang, Elixir, Scala]:
      i. **Erlang vs Scala**: wanted to learn a new programming paradigm - functional and as Internet says, it is better suited for working with many parallel tasks.
      ii. **Erlang vs Elixir**: since both languages work on the same virtual machine, I chose Erlang only because its syntax looked much "cleaner".
   b. **Erlang/OTP** - framework for creating distributed systems well suited to solving the problems from this lab. work and just interesting to study.
   c. **Rebar3** - an Erlang tool that makes it easy to compile and run project with a big amount of files.
   d. **jsx** - an erlang library for manipulating json.
2. **Actors:**
   a. **'Application'** - actor to start and stop 'Main Supervisor'.
   b. **'Main Supervisor'** - actor to start 'SSE listener Supervisor', 'Worker Supervisor', 'Worker Manager' and 'Worker Scaler'.
   c. **'SSE listener Supervisor'** - actor by initialization get a list of URLs and for every URL start a new 'SSE listener' actor.
   d. **'SSE listener'** - actor to establish a SEE connection, listen to all events and send json data to 'Worker Manager'.
   e. **'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'.
   f. **'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.
   g. **'Worker Supervisor'** - actor to dynamically balance the amount of 'Workers'.
   h. **'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.
3. **Endpoints:**
   a. **'Worker Manager':**
      i. Async: {tweet - atom, Tweet - json data string}
      Get json data to send it to one of 'Workers'.
   b. **'Worker Scaler':**
      i. Async: {inc - atom}
      Get signal to increment current counter.

c. **'Worker':**
   i. Async: {tweet - atom, Tweet - json data string}
      Process tweet.
d. **'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.

4. Communication Diagram: