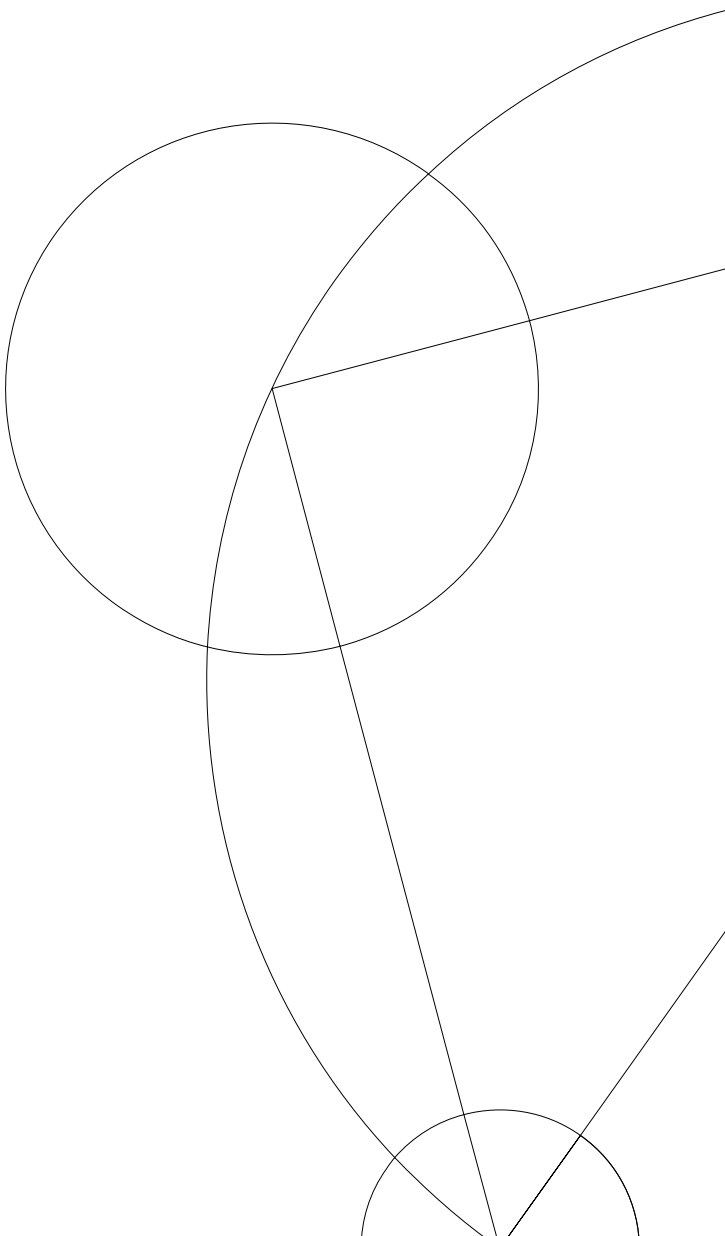
KØBENHAVNS UNIVERSITET

Home Assignment 4

Who Doesn't Love Emoji - Erlang

qbp758 / zxf339

AP22 Assignment 4 Group 10



Date: 7. oktober 2022

Part 1: Explain the Code Design

1. The main design of the code

There are five parts in emoji module:

1. There are 9 export functions in first part which can be invoked by other modules.
2. The second part is message transit process, which is used to receive and send messages in all module.
3. emoji server is the third part, it contains *loop* and *handle_call*.
4. In fourth part, there are a lot of helper functions for handling message in emoji server.
5. last, We design several basic functions or tool functions which support helper functions.

Short code snippets of five parts in emoji module are shown as follows.

```
% structure of emoji module
%% ====== part1:export functions
-spec start(initial()) -> any().
start(Initial) ->
    ...
%% ====== part2: message transit process
request_reply(Eid, Request) ->
    Ref = make_ref(),
    Eid ! {self(), Ref, Request},
    receive
        {_, Response} -> Response
    end.
%% ====== part3: emoji server
loop(State) ->
    receive
        ...
%% ====== part4: helper functions for handling messages
new_shortcode_helper({EmoList, Alias, Analytics}, Short, Emo) ->
    {NewEmoList, Res}=find_emo(EmoList, Short, Emo),
    ...
%% ====== part5:basic functions for helper function as above
find_emo(EmoList, Short, Emo) ->
    Exist = orddict:is_key(Short,EmoList),
    case Exist of
    ...
```

2. How we deal with unreliable analytics functions

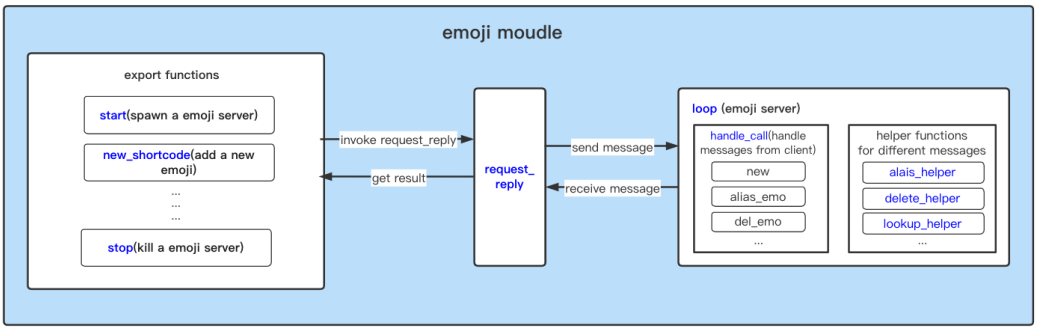
Unreliable analytics functions may throw errors or execute timeout in run_time. We do not care about the errors, we just update the origin state in analytics functions.

3. What state the server (and other relevant processes) maintain

There are two main processes in emoji module:

1. *request_reply* is a message transit process which sends and receives messages with *loop* (emoji server).
2. *loop* is the emoji server process which receives messages from *request_reply* and then handles it to send messages with result to *request_reply*.

A Diagram of the Processes



Part 2: Assessment of the Quality of Our code

A. Completeness

We have completed all the questions, although we spent a bunch of time,but way less than last time.

B. Correctness

Our code has passed the unit-testing and almost passed the onlineTA-testing.

1. There is only one failed in the OnlineTA-testing as below, but we indeed pass it in the local-testing.

   ```
   Register Hit and Last and check that they work with aliases:  ... *FAILED*
   ```

   NB: We have asked one TA, he didn't know how to fix it and he told us: our code makes sense. So we did a local-testing again, fortunately, it pass finally. (Please take a detailed look at the zip file "love_emoji.erl")

   ```
   % code snippets of local testing
      set_up() ->
          {ok, E} = emoji:start([]),
          ok = emoji:new_shortcode(E, "poop", <<"\xF0\x9F\x92\xA9">>),
          ok = emoji:alias(E, "poop", "hankey"),
          ok = emoji:alias(E, "hankey", "hankey1"),
          ok = emoji:analytics(E, "poop", fun hit/2, "Counter", 0),
          ok = emoji:analytics(E, "poop", fun accessed/2, "Accessed", []),
          ...
      try_it() ->
          {ok, Stats1} = emoji:get_analytics(E, "poop"),
          {ok, Stats2} = emoji:get_analytics(E, "hankey"),
          {ok, Stats3} = emoji:get_analytics(E, "hankey1"),
          ...
   ```

   We get right result from out local testing, the result is shown as follows.

Result of Local Testing

```
Poppy statistics:
  Accessed: [{"poop",{{2022,10,7},{12,9,6}}}]
  Counter: 1
hankey statistics alias:
  Accessed: [{"poop",{{2022,10,7},{12,9,6}}}]
  Counter: 1
hankey1 statistics alias:
  Accessed: [{"poop",{{2022,10,7},{12,9,6}}}]
  Counter: 1
```

2. The descriptions of testing are both attached as the upload files:
   - The result of Unit-testing can be seen in the "unit_test.txt" file
   - The result of OnlineTA-testing can be seen in the "onlineTA_exec_result.txt" file

C. Efficiency

We did try our best to improve the efficiency of code operation and make efforts to improve space usage, and it turns out that our efforts paid off.

D. Robustness

As we described above, we design five parts in emoji module, including 9 export functions which can be invoked by other modules, which make our code perform robustness.

E. Maintainability

We both think the maintainability of our code is quite good. As we added enough common code to improve the readability of the code and abstracted functions to make our code logically clarified.