Concurrent programming

BENZA Amandine – FORNALI Damien

Chat Server in Elixir

Architecture

In order to achieve the chat server, we used the following approach.

When the server main processor [**MP**] launches, it creates a new processor that will handle the users idle timeout checking [**IDLE_CHECKER**].

After that, it generates a listener in order to listen to new user connections [**LISTENER**] and start a unique processor for user acceptation [**ACCEPTOR**].

ACCEPTOR will checks the user inquired pseudo and if it is wrong, it will ask to the user to retry. If the inquired pseudo is valid, a new processor responsible of the newly connected user will be generated [**USER_PROC**].

MP will then waits for signals and processing according what it receives. For example if **MP** receives a « **broadcast** » signal, it means it has to send to all users a message.

Communication

And this is how we perform communication between processors. Only MP stores the users in his memory. Other USER_PROCs don't know existence of each other. To be able to communicate between them, USER_PROCs have to send a signal to MP in order to perform a certain operation, for instance to communicate messages.

Data approach

A user is represented by the following structure.

```
defmodule User do
        defstruct socket: nil, pseudo: nil, pid: -1, idleTime: 0
end
```

It allows us to keep trace of user information such as pseudo and associated processor identifier.

Difficulties

It was quite difficult to create a functional architecture as *Elixir*' structures cannot be shared as global variables. Making use of Elixir structures has also been difficult when handling idle timeout force disconnections.