1st Project
Concurrent Programming

Luís Guimarães 201703832

March 2021

1 Erlang

Erlang is a concurrent, functional programming language developed by Joe Armstrong, Robert Virding, and Mike Williams at Ericsson in 1986.[1]

It was designed to be used in systems with the following traits[3]:

- distributed
- fault-tolerant
- highly available
- hot swapping

Erlang applications are built of lightweight Erlang processes. Some principles of these processes are[2]:

- everything is a process
- processes are strongly isolated
- message passing is the only way for processes to interact
- processes do what they are supposed to do or fail

2 Architecture

This project is built of 2 components: **client** and **server**.

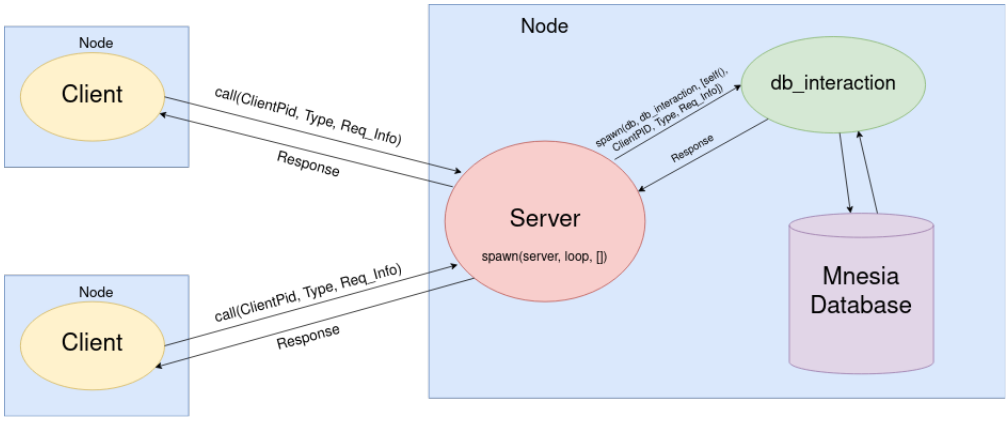

Figure 1: Architecture

The client consists of a function **call** that takes 3 arguments:

- **Node**, an atom that represents the server node (short name, ie: erl -sname server)
- **Type**, an atom that represents the request mode (lookup or update)
- **Req_Info**, a tuple that contains the request type (loans, books, return, etc.) and Info (either a single value or a tuple of values)

It sends a message to the process **server_pid** running at the given **Node** and outputs the **response** the server sends back.

In turn the server, is a more complex process. It consists of an infinite **loop** process that receives messages from the client, **spawns** a process **db_interaction** that communicates with the **mnesia** database.
This **db_interaction** process then sends a **response** to the server which is then relayed to the client.

3 Examples

3.1 Lookup

3.1.1 Books

Given a citizen card number, returns the list of books requested by that person.

```
client:call(server@localhost, lookup, {books, 123456789}).

Response: ["Brave New World","Women","Kafka on the Shore","1984"]
```

Figure 2: Lookup Books

3.1.2 Loans

Given a book title, returns the list of people who have requested that book.

```
client:call(server@localhost, lookup, {loans, "1984"}).

Response: ["David Bowie","Jean-Michel Jarre","Elton John"]
```

Figure 3: Lookup Loans

3.1.3 Requested

Given a book code, returns a boolean, **true** if it has been requested **false** if it has not.

```
client:call(server@localhost, lookup, {requested, 9780099458326}).

Response: true
```

Figure 4: Lookup Requested

3.1.4 Codes

Given a book title, returns a list of codes of books with that title.

```
client:call(server@localhost, lookup, {codes, "Brave New World"}).

Response: [9780060850523]
```

Figure 5: Lookup Codes

3.1.5 Number of Requests

Given a citizen card number, returns the number of books requested by that person.

```
client:call(server@localhost, lookup, {reqNum, 123456789}).

Response: [4]
```

Figure 6: Lookup Number of Requests

3.2 Update

3.2.1 Request

Given a person's data and a book code, adds the book to that person's requested books in the database.

```
client:call(server@localhost, update, {request, {123456790, 9780061177590}}).

Response: ok
```

Figure 7: Update Request

3.2.2 Return

Given a citizen card number and a book code, removes the book from that person's requested books in the database.

```
client:call(server@localhost, update, {return, {123456790, 9780061177590}}).

Response: ok
```

Figure 8: Update Return

References

[1] Joe Armstrong. *A history of Erlang, HOPL III: Proceedings of the third ACM SIGPLAN conference on History of programming languages*. Association for Computing Machinery, 2007. ISBN: 9781595937667. URL: https://dl.acm.org/doi/10.1145/1238844.1238850.

[2] Joe Armstrong. *Making reliable distributed systems in the presence of software errors*. The Royal Institute of Technology, Stockholm, Sweden, 2003. URL: http://erlang.org/download/armstrong_thesis_2003.pdf.

[3] Joe Armstrong et al. *Open-source Erlang – White Paper*. URL: https://web.archive.org/web/20111025022940/http://ftp.sunet.se/pub/lang/erlang/white_paper.html. (accessed: 28.03.2021).