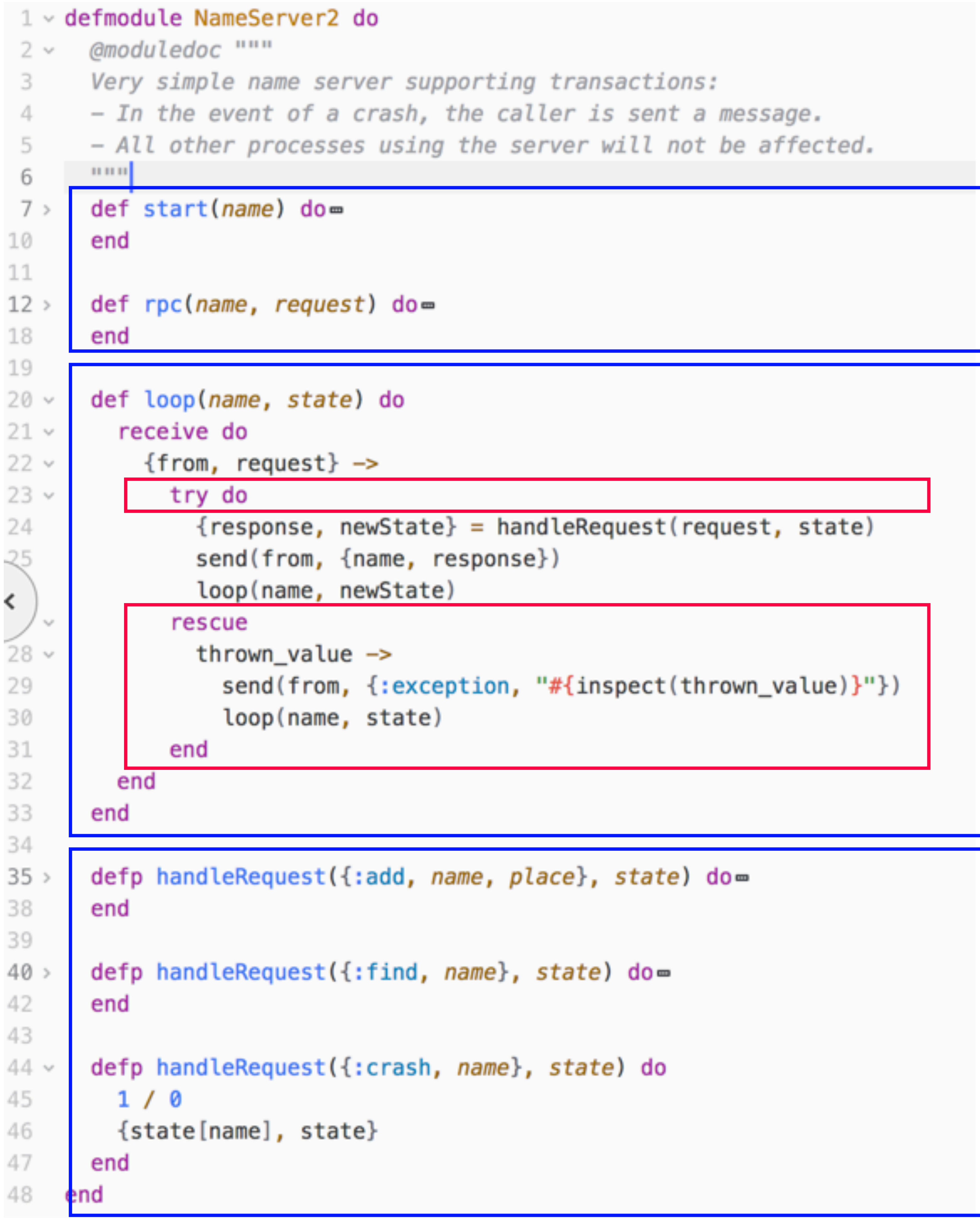


```elixir
defmodule NameServer2 do
  @moduledoc """
  Very simple name server supporting transactions:
  - In the event of a crash, the caller is sent a message.
  - All other processes using the server will not be affected.
  """
  def start(name) do
  end

  def rpc(name, request) do
  end

  def loop(name, state) do
    receive do
      {from, request} ->
        try do
          {response, newState} = handleRequest(request, state)
          send(from, {name, response})
          loop(name, newState)
        rescue
          thrown_value ->
            send(from, {:exception, "#{inspect(thrown_value)}"})
            loop(name, state)
        end
    end
  end

  defp handleRequest({:add, name, place}, state) do
  end

  defp handleRequest({:find, name}, state) do
  end

  defp handleRequest({:crash, name}, state) do
    1 / 0
    {state[name], state}
  end
end
```

Interface

Functions called by the process using the server.

Server

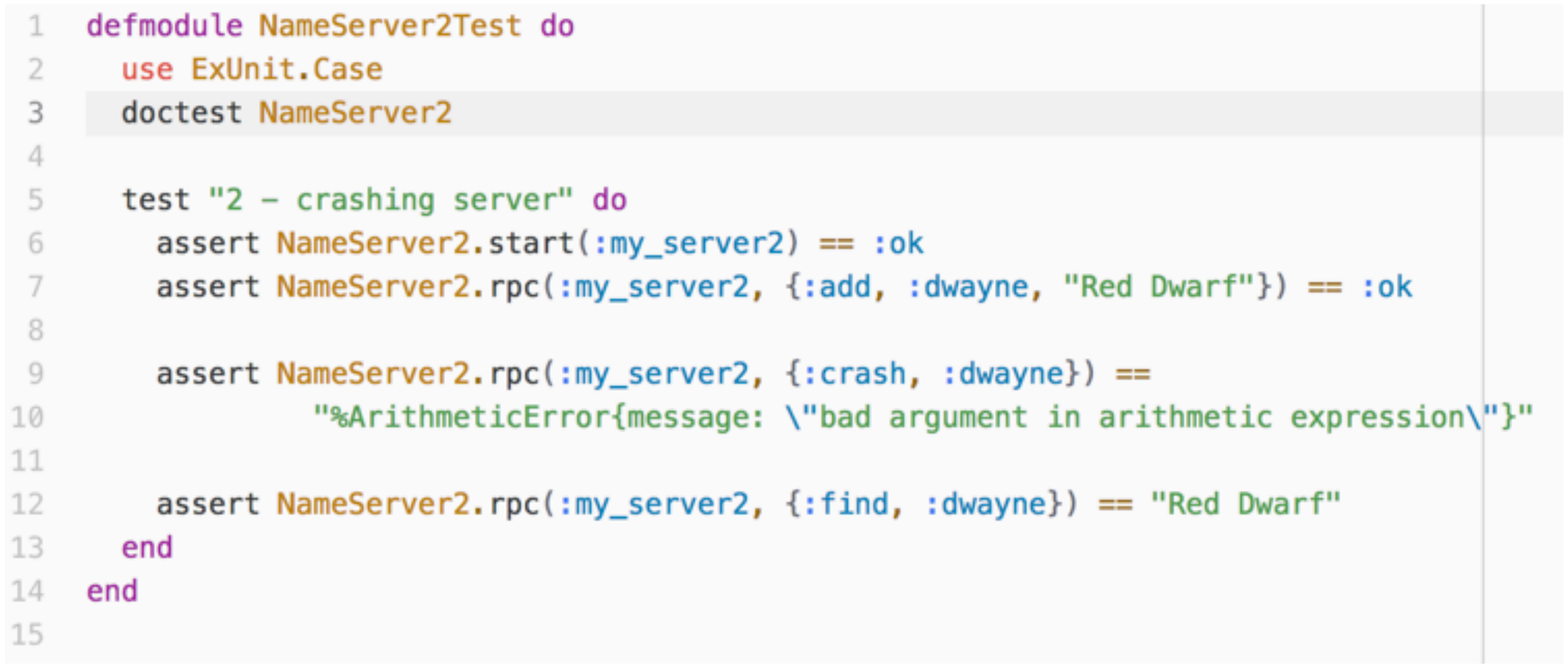
Runs in the server Process.

Implementation

Functions called by the server process to implement the server logic.

Nameserver2 Test

- Start the server.
- Add data.
- Crash the server.
- Check the data is still there.



```elixir
defmodule NameServer2Test do
  use ExUnit.Case
  doctest NameServer2

  test "2 - crashing server" do
    assert NameServer2.start(:my_server2) == :ok
    assert NameServer2.rpc(:my_server2, {:add, :dwayne, "Red Dwarf"}) == :ok

    assert NameServer2.rpc(:my_server2, {:crash, :dwayne}) ==
             "%ArithmeticError{message: \"bad argument in arithmetic expression\"}"

    assert NameServer2.rpc(:my_server2, {:find, :dwayne}) == "Red Dwarf"
  end
end
```