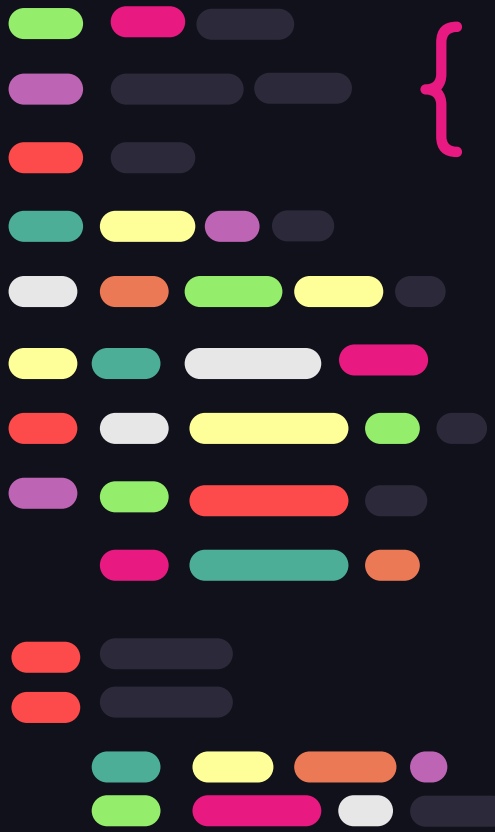
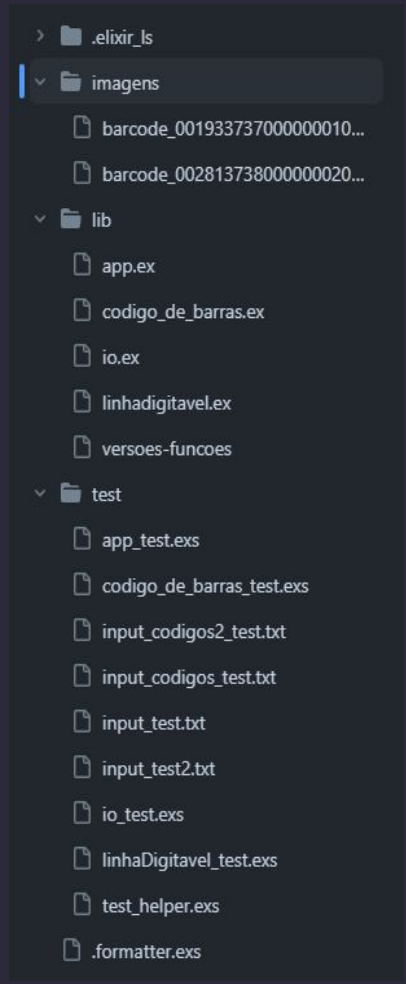
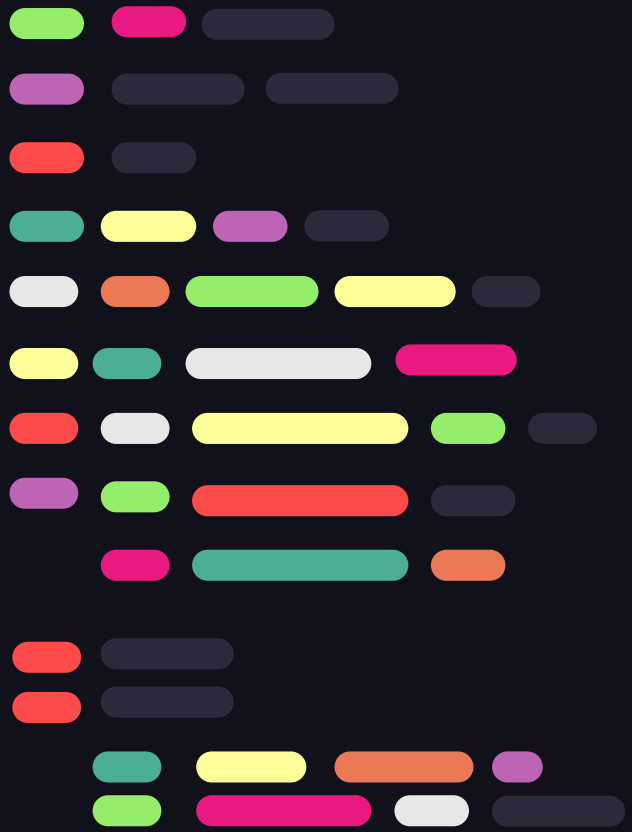


Codificador e decodificador de código de barras usando Elixir

< Cleiane Bondade >
< Gabriel da Costa >

Estrutura do Projeto





Testes automatizados

```
ubuntu@Gabriel:~/elixir/codigo-de-barras-elixir$ mix test
cb.gerarLinhaDigitavel: 0.029 ms
.cb.dvCodigoBarras: 0.003 ms
.cb.stringParaMoeda: 0.019 ms
.cb.toString: 0.005 ms
.cb.decodificar: 111.352 ms
.cb.gerar: 11.232 ms
.ld.toString: 0.009 ms
.ld.dvBloco: 0.006 ms
.ld.inserirDvs: 0.007 ms
.ld.gerar: 0.012 ms
.ld.ordenar: 0.002 ms
.fv.fatorVencimento: 0.787 ms
.fv.decode: 0.411 ms
.es.lerRegistros: 2.368 ms
.es.lerCodigos: 0.063 ms
.app.codificador: 16.832 ms
.app.decodificador: 2.861 ms
.

Finished in 0.2 seconds (0.00s async, 0.2s sync)
17 tests, 0 failures

Randomized with seed 479435
ubuntu@Gabriel:~/elixir/codigo-de-barras-elixir$
```

LinhaDigitavel

```elixir
defmodule LinhaDigitavel do
  def dvBloco(lista) do
    Enum.reverse(lista)
    |> Enum.map_every(2, fn x -> (x * 2) |> Integer.digits() |> Enum.sum() end)
    |> Enum.sum()
    |> (&(10 - rem(&1, 10))).()
  end

  def ordenar(codigodebarras) do
    codigodebarras
    |> Enum.slice(9..18)
    |> then(fn append -> Enum.slice(codigodebarras, 5..8) ++ append end)
    |> then(fn append -> Enum.slice(codigodebarras, 4..4) ++ append end)
    |> then(fn append -> Enum.slice(codigodebarras, 34..43) ++ append end)
    |> then(fn append -> Enum.slice(codigodebarras, 24..33) ++ append end)
    |> then(fn append -> Enum.slice(codigodebarras, 19..23) ++ append end)
    |> then(fn append -> Enum.slice(codigodebarras, 0..3) ++ append end)
  end

  def inserirDvs(lista) do
    lista
    |> Enum.slice(0..8)
    |> dvBloco()
    |> then(fn dv -> List.insert_at(lista, 9, dv) end)
    |> then(fn novaLista ->
      Enum.slice(novaLista, 10..19) |> dvBloco() |> (&List.insert_at(novaLista, 20, &1)).()
    end)
    |> then(fn novaLista ->
      Enum.slice(novaLista, 21..30) |> dvBloco() |> (&List.insert_at(novaLista, 31, &1)).()
    end)
  end

  def toString(lista) do
    lista
    |> List.insert_at(5, ".")
    |> List.insert_at(11, " ")
    |> List.insert_at(17, ".")
    |> List.insert_at(24, " ")
    |> List.insert_at(30, ".")
    |> List.insert_at(37, " ")
    |> List.insert_at(39, " ")
    |> Enum.join("")
  end

  def gerar(lista) do
    ordenar(lista)
    |> inserirDvs()
    |> toString()
  end
end
```

CodigoDeBarras

```elixir
defmodule CodigoDeBarras do
  defp mult(x) when x > 7, do: rem(x, 8) + 2
  defp mult(x), do: x + 2

  defp ajustarDv(x) when x >= 10, do: 1
  defp ajustarDv(x), do: x

  def dvCodigoBarras(lista) do
    Enum.reverse(lista)
    |> Enum.with_index(fn x, i -> x * mult(i) end)
    |> Enum.sum()
    |> (&(11 - rem(&1, 11))).()
    |> ajustarDv()
  end

  defp formatarValor(valor) do
    String.replace(valor, ",", "")
    |> String.pad_leading(10, "0")
  end

  def gerar(registro) do
    (registro.banco <>
       registro.moeda <>
       FatorVencimento.fatorVencimento(registro.dataVencimento) <>
       formatarValor(registro.valor) <> registro.convenio)
    |> String.graphemes()
    |> Enum.map(&String.to_integer(&1))
    |> then(fn lista -> List.insert_at(lista, 4, dvCodigoBarras(lista)) end)
  end

  def gerarLinhaDigitavel(codigo) do
    String.graphemes(codigo)
    |> Enum.map(&String.to_integer(&1))
    |> LinhaDigitavel.gerar()
  end

  defp ajustarMoeda(valor) do
    cond do
      String.length(valor) == 3 -> "0" <> valor
      String.length(valor) == 2 -> String.replace(valor, ",", "0,0")
      true -> valor
    end
  end

  def stringParaMoeda(valor) do
    String.to_integer(valor)
    |> Integer.to_string()
    |> String.split_at(-2)
    |> then(fn {inicio, fim} -> inicio <> "," <> fim end)
    |> ajustarMoeda()
  end

  def decodificar(codigo) do
    %{
      boleto: %{
        banco: String.slice(codigo, 0..2),
        moeda: String.slice(codigo, 3..3),
        dataVencimento: String.slice(codigo, 5..8) |> FatorVencimento.decode(),
        valor: String.slice(codigo, 9..18) |> stringParaMoeda(),
        convenio: String.slice(codigo, 19..43)
      },
      linhaDigitavel: gerarLinhaDigitavel(codigo),
      codigoDeBarras: codigo
    }
  end
end
```

FatorVencimento

```elixir
defmodule FatorVencimento do
  defp contarDias(num) when num > 8999, do: rem(num, 9000) |> contarDias
  defp contarDias(num), do: num + 1000

  defp ajustarDataFutura(data) do
    if Date.compare(data, Date.utc_today()) == :lt do
      Date.add(data, 9000)
      |> ajustarDataFutura()
    else
      data
    end
  end

  def fatorVencimento(data) do
    String.replace(data, "/", "-")
    |> Timex.parse!("{0D}-{0M}-{YYYY}")
    |> Date.diff(~D[2000-07-03])
    |> contarDias()
    |> Integer.to_string()
  end

  def decode(fator) do
    {_, data} =
      String.to_integer(fator)
      |> then(fn x -> Date.add(~D[2000-07-03], x - 1000) end)
      |> ajustarDataFutura()
      |> Timex.format("{0D}/{0M}/{YYYY}")

    data
  end
end
```

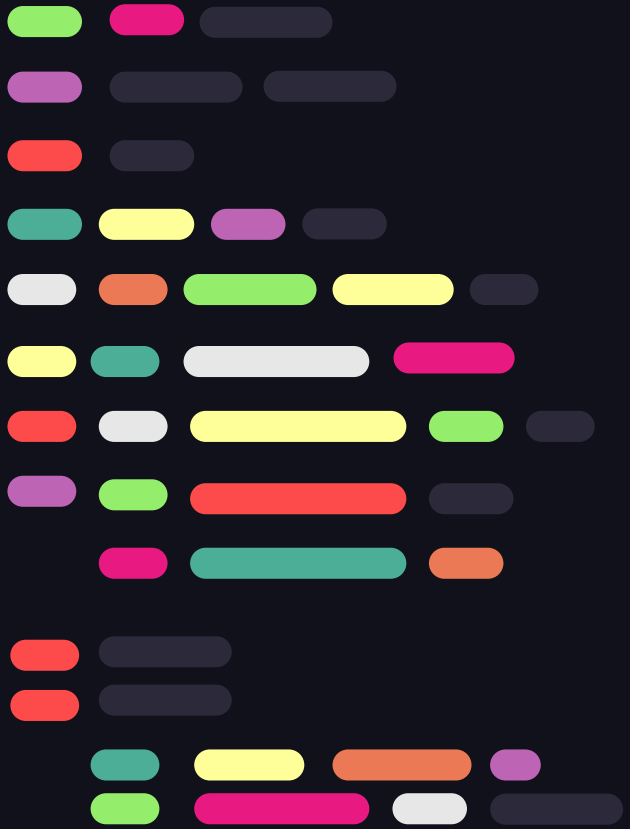
EntradaSaida

```elixir
defmodule EntradaSaida do
  def lerRegistros(path) do
    case File.read(path) do
      {:ok, conteudo} ->
        String.split(conteudo, "\n")
        |> Enum.chunk_every(5, 5, :discard)
        |> Enum.map(fn registro ->
          [banco, moeda, data, valor, convenio] = registro
          %{banco: banco, moeda: moeda, dataVencimento: data, valor: valor, convenio: convenio}
        end)

      {:error, _reason} ->
        IO.puts("Erro ao ler o arquivo")
        %{}
    end
  end

  def lerCodigos(path) do
    case File.read(path) do
      {:ok, conteudo} ->
        String.split(conteudo, "\n")

      {:error, _reason} ->
        IO.puts("Erro ao ler o arquivo")
        %{}
    end
  end
end
```

App

```elixir
defmodule App do
  defp salvarCodificacao(codigo, registro) do
    codigoFormatado = CodigoDeBarras.toString(codigo)

    Barlix.ITF.encode!(codigoFormatado)
    |> Barlix.PNG.print(file: "imagens/barcode_#{codigoFormatado}.png")

    %{
      boleto: registro,
      codigoDeBarras: codigoFormatado,
      linhaDigitavel: LinhaDigitavel.gerar(codigo)
    }
  end

  def codificador(arquivo_input) do
    EntradaSaida.lerRegistros(arquivo_input)
    |> Enum.map(fn registro -> CodigoDeBarras.gerar(registro) |> salvarCodificacao(registro) end)
  end

  def decodificador(arquivo_input) do
    EntradaSaida.lerCodigos(arquivo_input)
    |> Enum.map(fn codigo -> CodigoDeBarras.decodificar(codigo) end)
  end
end
```

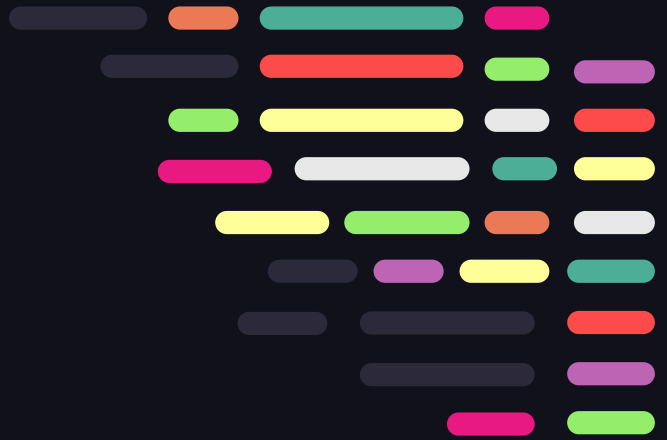
Testes (App)



```elixir
defmodule AppTest do
  use ExUnit.Case
  doctest App

  @registro1 %{
    banco: "001",
    convenio: "0500940144816060680935031",
    dataVencimento: "21/08/2032",
    moeda: "9",
    valor: "1,00"
  }
  @registro2 %{
    banco: "002",
    convenio: "0500940144816060680935031",
    dataVencimento: "22/08/2032",
    moeda: "8",
    valor: "2,00"
  }

  @mapa1 %{
    codigoDeBarras: "00193373700000001000500940144816060680935031",
    linhaDigitavel: "00190.50095 40144.816069 06809.350314 3 37370000000100",
    boleto: @registro1
  }
  @mapa2 %{
    codigoDeBarras: "00281373800000002000500940144816060680935031",
    linhaDigitavel: "00280.50094 40144.816069 06809.350314 1 37380000000200",
    boleto: @registro2
  }

  test "app.codificador" do
    {tempo_em_microssegundos, _} =
      :timer.tc(fn ->
        assert App.codificador("test/input_test.txt") == [@mapa1]
        assert App.codificador("test/input_test2.txt") == [@mapa1, @mapa2]
      end)

    tempo_em_milissegundos = tempo_em_microssegundos / 1000
    IO.puts("app.codificador: #{tempo_em_milissegundos} ms")
```

Otimização e legibilidade

```
20  ==========================================================================================
21  ========== primeira versão
22  def ordenar(codigodebarras) do
23          codigodebarras
24          |> Enum.slice(0..3)
25          |> Kernel.++(Enum.slice(codigodebarras, 19..23))
26          |> Kernel.++(Enum.slice(codigodebarras, 24..33))
27          |> Kernel.++(Enum.slice(codigodebarras, 34..43))
28          |> Kernel.++(Enum.slice(codigodebarras, 4..4))
29          |> Kernel.++(Enum.slice(codigodebarras, 5..8))
30          |> Kernel.++(Enum.slice(codigodebarras, 9..18))
31  end
32
33  ========== segunda versão com prepend
34  defp prepend(append,codigodebarras, intervaloInicial, intervaloFinal) do
35          Enum.slice(codigodebarras, intervaloInicial..intervaloFinal) ++ append
36  end
37
38
39  def ordenar(codigodebarras) do
40          codigodebarras
41          |> Enum.slice(9..18)
42          |> prepend(codigodebarras,5,8)
43          |> prepend(codigodebarras,4,4)
44          |> prepend(codigodebarras,34,43)
45          |> prepend(codigodebarras,24,33)
46          |> prepend(codigodebarras,19,23)
47          |> prepend(codigodebarras,0,3)
48  end
```



```elixir
def ordenar(codigodebarras) do
  codigodebarras
  |> Enum.slice(9..18)
  |> then(fn append -> Enum.slice(codigodebarras, 5..8) ++ append end)
  |> then(fn append -> Enum.slice(codigodebarras, 4..4) ++ append end)
  |> then(fn append -> Enum.slice(codigodebarras, 34..43) ++ append end)
  |> then(fn append -> Enum.slice(codigodebarras, 24..33) ++ append end)
  |> then(fn append -> Enum.slice(codigodebarras, 19..23) ++ append end)
  |> then(fn append -> Enum.slice(codigodebarras, 0..3) ++ append end)
end
```

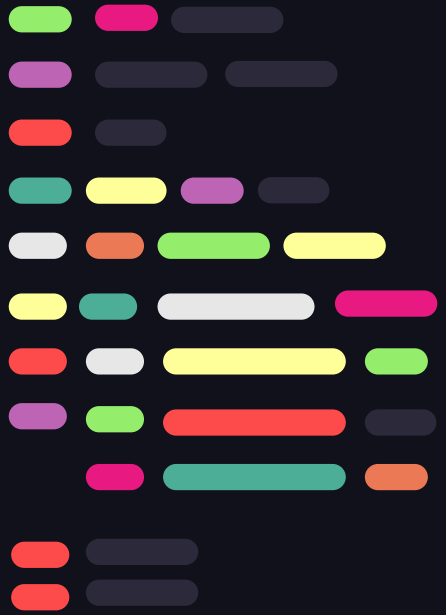
Otimização e legibilidade

```
~/Tarefa2/codigodebarra$ mix test
..LinhaDigitavel.ordenar primeira versão: 0.01 milissegundos
........
Finished in 1.1 seconds (0.00s async, 1.1s sync)
10 tests, 0 failures

Randomized with seed 982339
~/Tarefa2/codigodebarra$ mix test
LinhaDigitavel.ordenar primeira versão: 0.012 milissegundos
..........
Finished in 0.8 seconds (0.00s async, 0.8s sync)
10 tests, 0 failures

Randomized with seed 488057
~/Tarefa2/codigodebarra$ mix test
LinhaDigitavel.ordenar primeira versão: 0.009 milissegundos
..........
Finished in 0.9 seconds (0.00s async, 0.9s sync)
10 tests, 0 failures

Randomized with seed 682655
~/Tarefa2/codigodebarra$ mix test
..LinhaDigitavel.ordenar primeira versão: 0.009 milissegundos
........
Finished in 0.8 seconds (0.00s async, 0.8s sync)
10 tests, 0 failures

Randomized with seed 979036
~/Tarefa2/codigodebarra$ mix test
..LinhaDigitavel.ordenar primeira versão: 0.006 milissegundos
........
Finished in 0.7 seconds (0.00s async, 0.7s sync)
10 tests, 0 failures

Randomized with seed 884844
~/Tarefa2/codigodebarra$ mix test
..LinhaDigitavel.ordenar primeira versão: 0.012 milissegundos
........
Finished in 0.8 seconds (0.00s async, 0.8s sync)
10 tests, 0 failures

Randomized with seed 182313
~/Tarefa2/codigodebarra$ mix test
LinhaDigitavel.ordenar primeira versão: 0.009 milissegundos
```

```
~/Tarefa2/codigodebarra$ mix test
Compiling 1 file (.ex)
..LinhaDigitavel.ordenar segunda versão: 0.006 milissegundos
........
Finished in 1.0 seconds (0.00s async, 1.0s sync)
10 tests, 0 failures

Randomized with seed 657076
~/Tarefa2/codigodebarra$ mix test
.LinhaDigitavel.ordenar segunda versão: 0.006 milissegundos
.........
Finished in 1.2 seconds (0.00s async, 1.2s sync)
10 tests, 0 failures

Randomized with seed 23804
~/Tarefa2/codigodebarra$ mix test
LinhaDigitavel.ordenar segunda versão: 0.005 milissegundos
..........
Finished in 0.9 seconds (0.00s async, 0.9s sync)
10 tests, 0 failures

Randomized with seed 919775
~/Tarefa2/codigodebarra$ mix test
.LinhaDigitavel.ordenar segunda versão: 0.005 milissegundos
.........
Finished in 0.9 seconds (0.00s async, 0.9s sync)
10 tests, 0 failures

Randomized with seed 324759
~/Tarefa2/codigodebarra$ mix test
LinhaDigitavel.ordenar segunda versão: 0.008 milissegundos
..........
Finished in 1.0 seconds (0.00s async, 1.0s sync)
10 tests, 0 failures

Randomized with seed 233142
~/Tarefa2/codigodebarra$ mix test
..LinhaDigitavel.ordenar segunda versão: 0.006 milissegundos
........
Finished in 0.9 seconds (0.00s async, 0.9s sync)
10 tests, 0 failures

Randomized with seed 15860
~/Tarefa2/codigodebarra$ mix test
LinhaDigitavel.ordenar segunda versão: 0.006 milissegundos
```

```
~/Tarefa2/codigodebarra$ mix test
..LinhaDigitavel.ordenar usando prepend: 0.013 milissegundos
........
Finished in 0.9 seconds (0.00s async, 0.9s sync)
10 tests, 0 failures

Randomized with seed 24183
~/Tarefa2/codigodebarra$ mix test
LinhaDigitavel.ordenar usando prepend: 4.161 milissegundos
..........
Finished in 0.9 seconds (0.00s async, 0.9s sync)
10 tests, 0 failures

Randomized with seed 123081
~/Tarefa2/codigodebarra$ mix test
..LinhaDigitavel.ordenar usando prepend: 0.008 milissegundos
........
Finished in 0.9 seconds (0.00s async, 0.9s sync)
10 tests, 0 failures

Randomized with seed 418783
~/Tarefa2/codigodebarra$ mix test
..LinhaDigitavel.ordenar usando prepend: 0.012 milissegundos
........
Finished in 1.0 seconds (0.00s async, 1.0s sync)
10 tests, 0 failures

Randomized with seed 458654
~/Tarefa2/codigodebarra$ mix test
..LinhaDigitavel.ordenar usando prepend: 0.007 milissegundos
........
Finished in 0.8 seconds (0.00s async, 0.8s sync)
10 tests, 0 failures

Randomized with seed 117588
~/Tarefa2/codigodebarra$ mix test
LinhaDigitavel.ordenar usando prepend: 69.977 milissegundos
..........
Finished in 0.9 seconds (0.00s async, 0.9s sync)
10 tests, 0 failures

Randomized with seed 936840
~/Tarefa2/codigodebarra$ mix test
.LinhaDigitavel.ordenar usando prepend: 9.25 milissegundos
```

Limitações



Decodificação de imagem

Dados de convênio

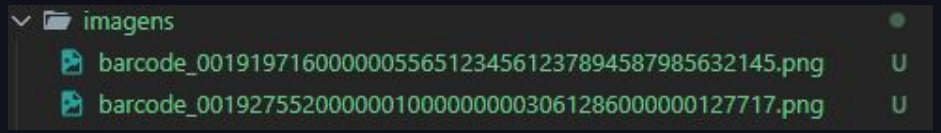
Valores com 10+ dígitos

Dados de entrada

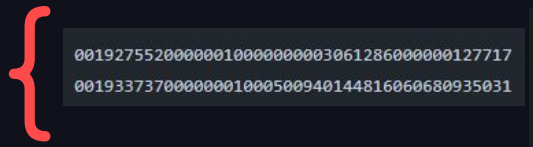
Exemplo de execução (codificador)

```
001
9
14/05/2024
55,65
12345612378945879856321 45
001
9
31/01/2043
10,00
00000030612860000001277 17
```

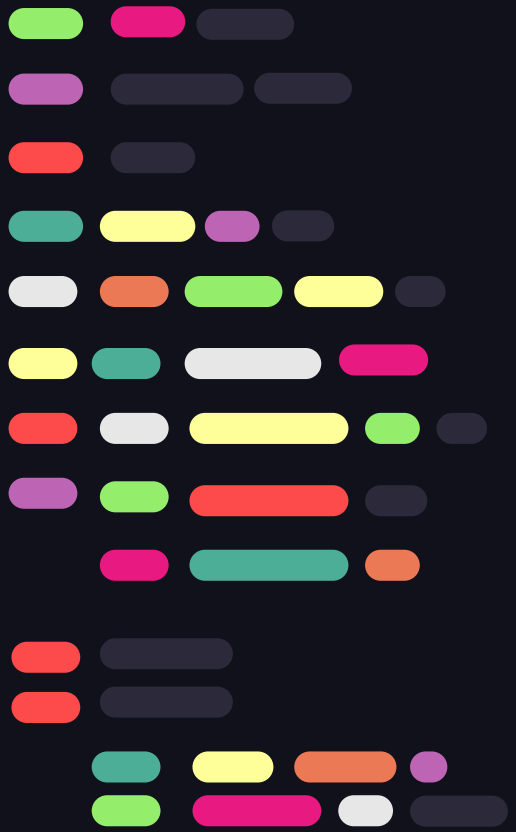
```
iex(4)> App.codificador("input.txt")
[
  %{
    boleto: %{
      banco: "001",
      convenio: "12345612378945879856321 45",
      dataVencimento: "14/05/2024",
      moeda: "9",
      valor: "55.65"
    },
    codigoDeBarras: "00191971600000055651234561237894587985632145",
    linhaDigitavel: "00191.23454 61237.894581 79856.321454 1 97160000005565"
  },
  %{
    boleto: %{
      banco: "001",
      convenio: "00000030612860000001277 17",
      dataVencimento: "31/01/2043",
      moeda: "9",
      valor: "10.00"
    },
    codigoDeBarras: "00192755200000010000000003061286000000127717",
    linhaDigitavel: "00190.00009 03061.286005 00001.277177 2 75520000001000"
  }
]
iex(5)>
```



Exemplo de execução (decodificador)




```
iex(4)> App.decodificador("decode.txt")
[
  %{
    boleto: %{
      banco: "001",
      convenio: "000000306128600000127717",
      dataVencimento: "31/01/2043",
      moeda: "9",
      valor: "10,00"
    },
    codigoDeBarras: "00192755200000010000000003061286000000127717",
    linhaDigitavel: "00190.00009 03061.286005 00001.277177 2 75520000001000"
  },
  %{
    boleto: %{
      banco: "001",
      convenio: "05009401448160606809350 31",
      dataVencimento: "21/08/2032",
      moeda: "9",
      valor: "1,00"
    },
    codigoDeBarras: "00193373700000001000500940144816060680935031",
    linhaDigitavel: "00190.50095 40144.816069 06809.350314 3 37370000000100"
  }
]
iex(5)>
```



Thanks!

CREDITS: This presentation template was created by **Slidesgo**, and includes icons by **Flaticon**, and infographics & images by **Freepik**