PRADITA University

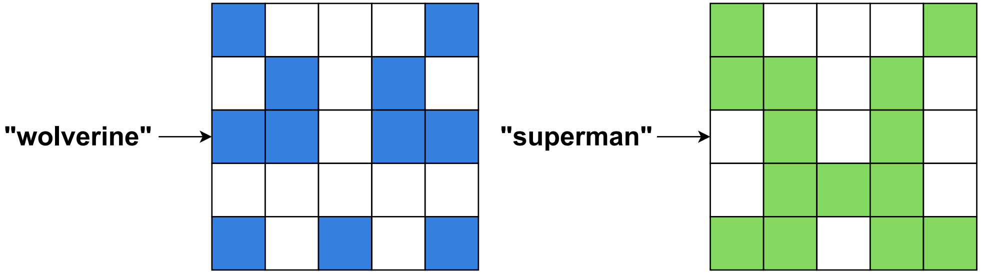
IF140303-Web Application Development

Session-05: Avatar Generator in Elixir

PRU/SPMI/FR-BM-18/0222

Alfa Yohannis

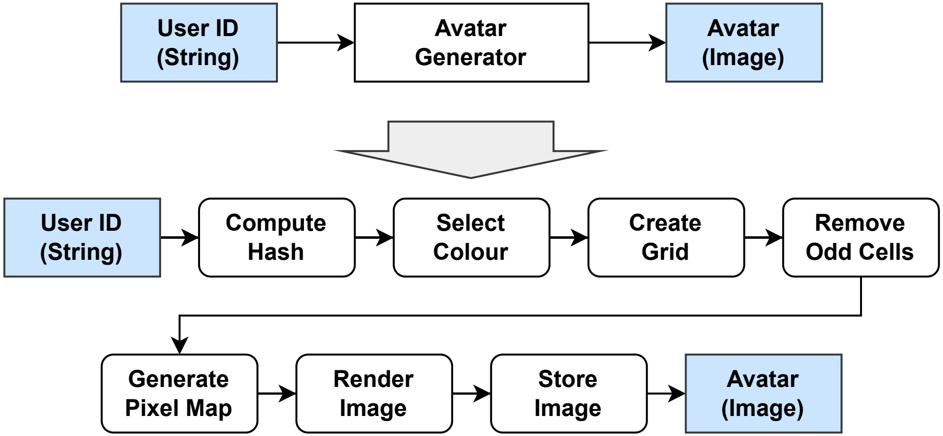
Avatar Generator



wolverine's md5 hash = [54, 129, 223, 141, 4, 71, 14, 204, 101, 5, 59, 121, 14, 25, 160, 101]

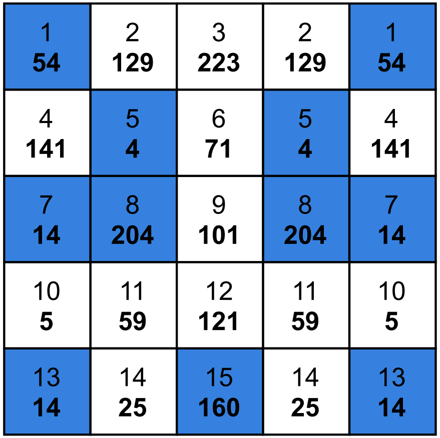
superman's md5 hash = [132, 217, 97, 86, 138, 101, 7, 58, 59, 207, 14, 178, 22, 178, 165, 118]

Avatar Pipeline



Avatar Computation

PRADITA University



1 **54**	2 **129**	3 **223**	2 **129**	1 **54**
4 **141**	5 **4**	6 **71**	5 **4**	4 **141**
7 **14**	8 **204**	9 **101**	8 **204**	7 **14**
10 **5**	11 **59**	12 **121**	11 **59**	10 **5**
13 **14**	14 **25**	15 **160**	14 **25**	13 **14**

The first 3 values are the RGB colour

[54, 129, 223, 141, 4, 71, 14, 204, 101, 5, 59, 121, 14, 25, 160, 101]

Avatar Overview

PRADITA University

- The avatar is represented by a 5x5 grid, with each cell being 50x50 pixels.
- The grid is symmetrical, where the middle column acts as a mirror for the left and right sides.
- This grid-based avatar is a unique, visual representation of input data, often used as a personal identifier.

Project: Avatar Generator App

PRADITA University

- We will create an avatar generator that produces a consistent avatar for the same input string.
- The avatar will be a visual identifier generated uniquely from the string.
- The project will be built step-by-step, with each step implemented through specific functions.

Avatar Generator Workflow

PRADITA University

1. Accept an input string.
2. Generate an MD5 hash from the string.
3. Convert the hash into a list of numbers.
4. Choose a color based on the hash values.
5. Create a symmetrical grid based on these numbers.
6. Convert the grid into an image.
7. Save the generated image as a PNG file.

Hashing the Input String

```
iex> hash = :crypto.hash(:sha256, "banana")
<<180, 147, 212, 131, 100, 175, 228, 77, 17, 192, 22, 92,
    244, 112, 164, 22, 77,
30, 38, 9, 145, 30, 249, 152, 190, 134, 141, 70, 173, 227,
    222, 78>>

iex> :binary.bin_to_list(hash)
[180, 147, 212, 131, 100, 175, 228, 77, 17, 192, 22, 92,
    244, 112, 164, 22, 77,
30, 38, 9, 145, 30, 249, 152, 190, 134, 141, 70, 173, 227,
    222, 78]
```

- We compute the MD5 hash of the input string using `:crypto.hash/2`.
- The binary hash is then converted into a list of integers using `:binary.bin_to_list/1`.

Starting the Implementation

```
def compute_hash(input) do
hash = :crypto.hash(:sha256, input)
|> :binary.bin_to_list

%Avatar.Image{hash: hash}
```

- We start by defining the `hash_input/1` function.
- This function takes a string, computes its hash, and returns a structure with the hash.

Running the Code in IEx

- To execute the code, we use IEx (Interactive Elixir) as follows:

```
iex> AvatarGenerator.generate("banana")
hash: [180, 147, 212, 131, 100, 175, 228, 77, 17, 192, 22,
    92, 244, 112, 164,
22, 77, 30, 38, 9, 145, 30, 249, 152, 190, 134, 141, 70,
    173, 227, 222, 78],
```

- This command will generate an avatar based on the string `"banana"`.

Generating RGB Values

- The first three values from the hash list are used to generate an RGB color.
- This color will be applied to specific cells in the grid.

Grid Pattern

- The grid pattern for the avatar follows this symmetry:

```
 1  2  3  2  1
 4  5  6  5  4
 7  8  9  8  7
10 11 12 11 10
13 14 15 14 13
```

- All even-numbered cells in the grid will be filled with the generated RGB color.

Summary

- We discussed the Avatar Generator project and its workflow.
- We covered how to hash a string and use the hash to determine the avatar's color.
- We also explained how the grid is formed and how colors are applied based on hash values.