HOMEWORK 06 - ROSHAN POUDEL

2. Breakdown of the Pipeline in `HelloCrawler.handle_response`

```
[
      url
      | body
        |> Floki.find("a")
        |> Floki.attribute("href")
        |> Enum.map(&URI.merge(url, &1))
        |> Enum.map(&to_string/1)
        |> Enum.reject(&Enum.member?(path, &1))
        |> Enum.map(&get_links(&1, [&1 | path], context))
        |> List.flatten()
    ]
```

1. **`Floki.find("a")`**:
   - Finds all `<a>` anchor tags in the HTML `body`.
   - Returns a list of Floki elements representing each anchor tag.
2. **`Floki.attribute("href")`**:
   - Extracts the `href` attribute from each anchor tag.
   - Returns a list of URLs as strings.
3. **`URI.merge(url, &1)`**:
   - Converts relative URLs into absolute ones by merging them with the base URL.
   - Returns a list of `URI` structs.
4. **`to_string/1`**:
   - Converts each `URI` struct to its string representation.
   - Returns a list of URLs as strings.
5. **`Enum.reject(&Enum.member?(path, &1))`**:
   - Filters out URLs already in the `path` (visited URLs).
   - Returns a list of new URLs to crawl.
6. **`Enum.map(&get_links(&1, [&1 | path], context))`**:
   - Recursively crawls each new URL and adds the crawled URLs to the list.
   - Returns a list of lists of crawled URLs.
7. **`List.flatten`**:
   - Flattens the list of lists into a single list of URLs.
   - Returns a list of crawled URLs.

Final Output:

- The result is a list of unique URLs found on the page, including any new links found from the `href` attributes of `<a>` tags.

3. HelloCrawler.get_links("http://scarl.sewanee.edu")

```
url_lists = HelloCrawler.get_links("http://scarl.sewanee.edu")
Crawling "http://scarl.sewanee.edu"...
["http://scarl.sewanee.edu",
 "https://new.sewanee.edu/programs-of-study/math-cs/",
 "http://www.sewanee.edu/", "http://scarl.sewanee.edu/CS270/",
 "http://scarl.sewanee.edu/CS326/",
 "http://scarl.sewanee.edu/other-classes.html",
 "http://dokuwiki.sewanee.edu/doku.php?id=computer_science_research",
 "http://scarl.sewanee.edu/Research/phobos.html",
 "http://scarl.sewanee.edu/cv.html",
"http://scarl.sewanee.edu/future.html",
 "http://www.cra.org/", "http://www.acm.org/", "http://ccscse.org/",
 "http://www.nealstephenson.com/reamde.html",
 "https://www.sewanee.edu/student-life/sewanee-outing-program/",
 "https://bandcamp.com/spc", "http://mob.rice.edu/",
 "http://scarl.sewanee.edu/statement.html",
 "https://www.timeanddate.com/worldclock/fullscreen.html?n=171",
 "https://circleid.com/posts/20221119-in-memoriam-frederick-p-brooks-jr-a-
personal-recollection",
 "https://amturing.acm.org/award_winners/brooks_1002187.cfm",
 "https://e-catalog.sewanee.edu/arts-sciences/departments-
interdisciplinary-programs/mathematics-computer-science/computer-science-
major/",
 "https://e-catalog.sewanee.edu/arts-sciences/departments-
interdisciplinary-programs/mathematics-computer-science/computer-science-
minor/",
 "https://warren.sewanee.edu/linuxlabs/",
 "https://sewanee.campuslabs.com/engage/organization/computingsociety",
 "https://x.com/sewaneecs"]

iex(6)> length(url_lists)
26
```

4. Compare running the sequential version versus using Tasks to process each URL

- Sequential Version

```
iex(2)> {time, list} = :timer.tc(HelloCrawler, :get_links,
["http://scarl.sewanee.edu"])
#displays the list of the urls visited
iex(3)> time
155313 #in microseconds
```

```
iex(4)> length(list)
26
```

- Tasks Version

```
ex(2)> {time, list} = :timer.tc(HelloCrawler, :get_links,
["http://scarl.sewanee.edu"])
#displays the list of the urls visited
iex(3)> time
156214
iex(4)> length(list)
26
```

In my case, both versions returned same number of links (26) but the sequential was faster than the tasks which is unexpected. One reason why this might happen is that we are visiting the link which does not have a lot of nested links. I trid crawling a new link "http://new.sewanee.edu" and the sequential took 112355 microseconds and the tasks version took 110265 microseconds which is as per our expectation.

5. Compute an estimate of the average time taken to fetch and process a linked webpage

156214 microseconds / 1_000_000 = 0.156 0.156 / 26 = 0.006 pages/second

6. Use HelloCrawler on some domain other than http://scarl.sewanee.edu.

- Used http://new.sewanee.edu
- 88 links got
- 58 links are from the target domain
- Used "https://warren.sewanee.com/scarl/CS157/" and got the same link back. I think it is because there are not links to crawl

Agents

1. The Agent in concurrent_search.exs is a lightweight and efficient tool for managing the shared state (a list of search results) in a concurrent Elixir program. It simplifies the process of aggregating results from multiple spawned processes, while maintaining safety and readability.

2. Done!

Genservers

1. Please find the refactored code in the metex folder