Phase 1

□ Check each link on the landing page and verify if any product or brand matches the offer description

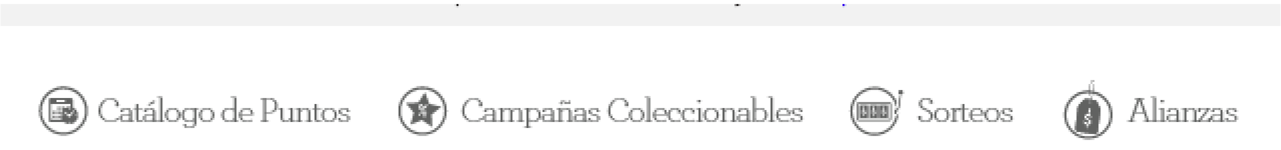
□ verifying if the link is taggead by making sure if this link has "utm"
keep in mind:

1. How scraping works
2. execution time
3. link status
4. scalable stack, (e.g., playwright, python, javascript)

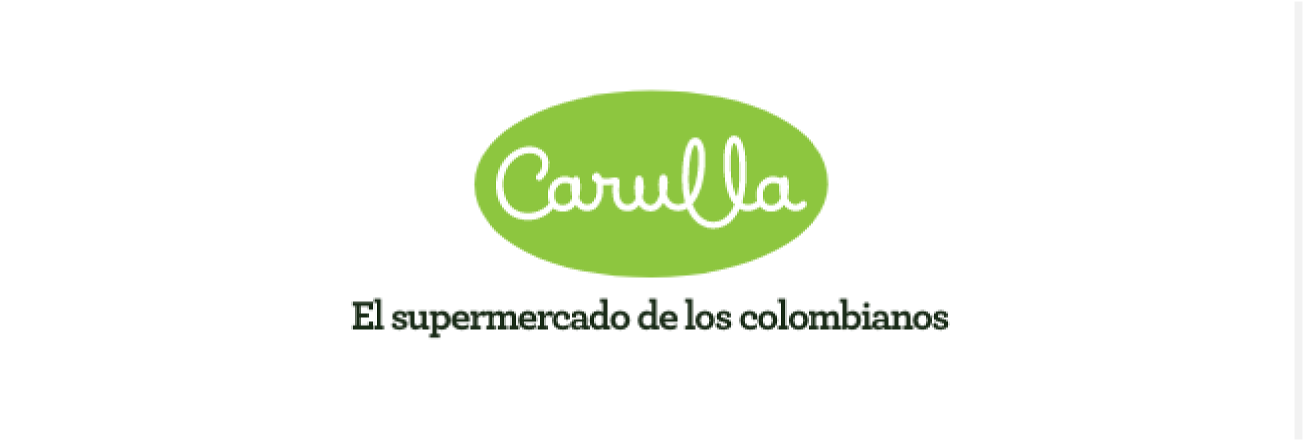
Test 1 - scrapy:

- Documentation: https://docs.scrapy.org/en/latest/topics/spiders.html

Improvements for the HTML structure:
Development requires to stablishing specific identifiers to enable link-search automation in the index provided by Dani:
Define classes to indicate whether an element is an offer or not:
1. no neccesary information like:



2. Header



3. Banner but this is relative
4. Canal footer

Nota: In the future this may not work, if something change, so the idea to classify those id or classs for a better search in the website

- ☐ Review the legal section for mistakes (e.g., repetitive sentences, incorrect dates -- For instance, if the date says '19 sept', and the current date is ‘20 Sept,’ it should be flagged (marcado)
  keep in mind:
  1. There's a figma design where this entire information comes from
  2. Each offer should be compared with the data in the excel file. However, the numbering sometimes is out of order, so an additional step is needed to ensure the numbering is correct

Random information:
I received a link that is composed of images within an HTML structure

Automate the QA mails process by verifying links and comparing the information data in an excel file

Phase 2 coming up soon

- create an image rag
- identify patterns
- usually mistakes
- tokens consumption
- include OCR processing internally, if required.
- optimization
- should it be a chat? or an endpoint?
- execution time
- will each image need a description?
- ideal stack technology