Build an Open Graph previewer

Task

Create a page that allows users to test and preview an Open Graph image from any given URL.

Context

Facebook's Open Graph protocol is a standard that enables web pages to be displayed richly on most major social media platforms. An Open Graph-compliant web page should include `<meta>` tags in its `<head>` to specify a variety of properties, including an image URL that can be used to display a preview of the web page. You may have seen these preview "cards" on Twitter, Medium, or Slack.

Here is an example of how those `<meta>` tags might look in the page source:

```
<html prefix="og: http://ogp.me/ns#">
<head>
  <title>Luna Care: On-Demand Physical Therapy</title>
  <meta property="og:title" content="Luna Care - Physical therapy,
  delivered to you">
  <meta property="og:type" content="website">
  <meta property="og:url" content="https://www.getluna.com/">
  <meta property="og:image"
  content="https://www.getluna.com/assets/images/we_come_to_you.svg">
  ...
</head>
...
</html>
```

Task Implementation Details

1. Create a new Rails app with a page where the user can enter any URL. This will mean having a controller file with at least one action to receive and kick off the URL processing, a view file for the page, and whatever you want for styling the page.

2. Process the URL, looking for its Open Graph `<meta>` tag with the `og:image` property. For simplicity you are free to assume there is at most one image for a given url.

   Important: Ensure that processing the URL is asynchronous to the Rails controller action. We do not want to block the web thread with external requests to user-submitted URLs.

3. Store the URL and its processing status so that your frontend previewer can display it.
4. When processing is finished, display the image to the user on the same page where they entered the URL.

We are looking more for thought process in terms of decision making around trade-offs than any specific implementation. Feel free to use any project generators or external gem or JavaScript libraries, including any flavor of JavaScript you want. The frontend status updating logic in particular may be done however makes sense to you given the time expectations of this project.

Please note that we use Postgres at Luna, so that would be the preferred choice of a data store. If you happen to build an app requiring a different data store, please make it easy to set up with some containerized environment. If no persistent datastore is used (e.g. Redis only) please explain the reasoning for your decision in more detail in the README.

Submission

Please either give us access to a private repository hosted on GitHub or your favorite git hosting service, or just send over a zip of your work that includes the .git/ directory.

Pre-submission checklist for your implementation:

1. A README that describes your approach, and how to set up and run your application.
2. The git history for your work. Please include the .git/ directory since that is what holds the git history.
3. Can your thought process be understood from the git history and README? Will this be easy for the reviewer(s) to set up?

What we're looking for:

- Clean, easy-to-read, and easy-to-understand code and README
- Efficient, thoughtful implementations. If anything is not production-worthy (or otherwise less than ideal) please call it out in the README.

IMPORTANT: The amount of effort expected to complete this task can range anywhere from 2 hours (*extremely* familiar with all the technologies involved) to 6 hours (completely unfamiliar with any of the technologies involved), maybe even a little more if you are taking it easy.

If you are going above and beyond for the sake of learning, that's great, but is definitely not expected nor required. We want to be respectful of your time, and we also want to treat all candidate submissions equally. Therefore we strongly encourage you to timebox working on this project to 4-6 hours, treating it as a mini "hackathon" session (please do not spend more than a full weekend day on this!). If you work considerably more time than expected, call it out in the README.

Also keep in mind this is for showcasing your strengths. If you are more backend-heavy, focus on making a great backend, if you are frontend-heavy, put your efforts into getting the backend working and making the frontend lovely. Just make sure everything is to spec from the Task Implementation Details section (no skipping steps please).

Lastly...
Have fun!!