HACKER NEWS AGGREGATOR

The Hacker News aggregator fetches data from the Hacker News API for further analysis or consumption.

The api documentation is available at: https://github.com/HackerNews/API.

FEATURES

- Every 5 minutes fetch the 50 top stories (polling the HN api)
- Make stories available via two public APIs: JSON over http and JSON over WebSockets (more details below)

PUBLIC APIs

- The http api should provide a way to list stories with pagination (10 results per page)
- The http api should provide a way to fetch a single story
- Upon connection, The WebSockets api should send a the 50 top stories. When stories get refreshed, new stories should be sent.

ESSENTIAL CONSTRAINTS

- All components need to be properly supervised
- All data should be kept memory (e.g. no external databases)

NICE TO HAVE

- Non blocking operations: for example fetching new stories should not affect the ability to read data currently in memory.
- Testing: a strategy for testing the single components with some examples (it's fine to leave some tests as pending as long as they explain what they would test).
- Type and function specifications.

REVIEW

ESL will review your implementation considering the following criteria:

- Resilience (e.g. what happens if the HN api is down?)
- Performance (e.g. are there bottlenecks that can be identified without even load testing the application?)
- Clarity (e.g. can I just open the Observer application and get a good insight on the application structure? Is there clear intent in function names?)
- Security (e.g. are there ways that the public API could be abused?)
- OTP usage (e.g. does this component need to be a supervised worker or is it more appropriate to just spawn a process?)
- Usage of dependencies (e.g. is it worth pulling in an entire package just for a single function?)
- Commit history (e.g. are commit messages clear? does each commit represent a unique and individual change?)

MORE THAN ONE WAY TO DO IT

Some of the features imply choosing among tradeoffs in consistency, performance and resilience. In those instances, feel free to leave a comment in the code to show that you're aware of them. The point of this exercise is to highlight your ability to reason around constraints.