smartlook

Backend Test Project

Create a Node.js API which will allow users to create and maintain custom collections of Hacker News stories. **Stories** with **comments** and the **story collections** should be persisted in a local SQL database. Stories should be fetched from HackerNews/API and story collections should be maintained through the API.

Bonus 1 - Access to the API should be secured with JWT token.

Bonus 2 - Index stories in Elasticsearch and use it’s full text search capabilities in a search API endpoint.

Bonus 3 - Regularly sync imported stories to keep them always up to date.

You can use any tools and libraries you know/like/find. We have only one strict requirement - use TypeScript. As a web framework we prefer Koa.js but feel free to use whichever one makes your ❤ pound.

Expected outputs

- Git repository with your source code, can be a public repo on GitHub, BitBucket, GitLab or just send us an archive.
- Automated test suite covering the basic API functionality.
- Readme with all the required steps to run the application and tests.

API Endpoints

1) Story collections endpoints (CRUD)

The concept of collections is not present on Hacker News and it's an extra feature provided by this project. It's maintained through the API and persisted in the local SQL database. A story collection entity should be really simple, it could for example contain only id, name and owner id.

2) Add story(ies) to the collection

Adds a HN story to a collection and persists a snapshot of the conversation fetched from the https://github.com/HackerNews/API in the database. You don't have to sync all the fields, just the most important ones (eg. time, content, author). Be careful here - fetching the whole story with its comments may take a while and it would be nice to process this task asynchronously.

3) Authorize user (optional)

Creates a new user if needed and returns a JWT token that has to be used to authenticate requests to all other endpoints.

4) Search (optional)

Perform a full text search in Elasticsearch through the stories and their comments in a specified collection or all collections that are owned by the authorized user.

Tips & tricks

1) Asynchronous Flow Control

Please make sure you understand all the standard ways of dealing with async flow control (callbacks, promises, async functions with await, event emitters). We strongly recommend you to use async/await in your solution.

2) API design

Although we like GraphQL (we even use it internally) we decided that we want you to create a REST API in this task. Please follow it's best practices and conventions.

3) Docker

Run any dependencies in Docker to make it simple for us to run your project locally. If you are not familiar with Docker, you can use this Docker Compose file to start Postgresql and

Elasticsearch and access them locally on localhost:5432 and localhost:9200. You can configure the application itself and the test suite to run in Docker container too but it’s not required.

```
# copy content to docker-compose.yml file
# run `docker-compose up` to start both containers, ctrl+c to stop them
# run `docker-compose down -v` to remove containers and docker volumes

version: "3.8"

services:
  postgres:
    image: postgres:13-alpine
    container_name: sl-postgres
    environment:
      POSTGRES_USER: hacker_news_stories
      POSTGRES_DB: hacker_news_stories
      POSTGRES_PASSWORD: hacker_news_stories
    volumes:
      - sl-postgres:/var/lib/postgresql/data
    ports:
      - 5432:5432

  elastic:
    image: docker.elastic.co/elasticsearch/elasticsearch-oss:7.10.2
    container_name: sl-elastic
    volumes:
      - sl-elastic:/usr/share/elasticsearch/data
    environment:
      - ES_JAVA_OPTS=-Xms256m -Xmx256m
      - discovery.type=single-node
    ports:
      - 9200:9200

volumes:
  sl-elastic:
    driver: local
  sl-postgres:
    driver: local
```

4) General Node.js best practices

You may take a look at https://github.com/goldbergyoni/nodebestpractices where you will find tons of resources about Node.js.