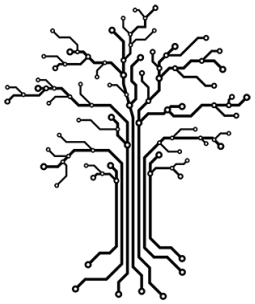
CLIMATETREE



STORIES MICROSERVICE

Developer Guide

Short Description

Stories are the posts users can view, create or delete from the climatetree.wiki which forms the crux of the project ClimateTree.
Stories microservice handles all the operations and storage of stories associated with the ClimateTree.
A story consists of all the data associated with the story, posted date of the story, rating, taxonomy information, informational links, user's information who created the story, comments, likes and flagged information for the story. Stories microservice stores all the stories in a MongoDB and consists of various APIs to query and transform data as requested by the frontend. Stories microservice also uses elastic search for faster retrieval and querying to help provide data quickly to the frontend for a better user experience.

Goals and Audience

The API Gateway is a direct audience of Stories microservice. But indirectly any user who logs into climatetree is an audience for the story that he/she views.

Tools, technologies, and servers used

1. Server - Node JS, Express
2. Database - MongoDB
3. Elasticsearch
4. In-memory database for unit tests - MongodbMemoryServer

API documentation link : Postman documentation link

Process when one pulls app from Git

Local database setup:
MongoDB local setup -

1. Download and install mongoDB from https://www.mongodb.com/
2. (Not very sure if this is needed) MongoDB needs a directory to store data, typically located at the root. a. mkdir /data b. mkdir /data/db c. On mac OS you may need to run with sudo
3. From command line use *'mongod'* to start the server(for mac you may need *'sudo mongod'*)
4. mongod server listens to port 27017

Test if mongodb is working -

1. On the command line, use *'mongo'* to connect to the database server.
2. In the mongodb command line you can use *'show dbs'* to print all the available databases. You should have a database called 'admin'

Add dummy data to mongoDB

1. Access the readme from here. Run the python script by following the instructions on the readme.md from the above link.
2. Run the *'show dbs'* command and you should see the climateTree database available

Setting the node server: Git repo link - https://github.com/climatetree/stories-microservice

1. Clone the repository in your local machine
2. Open terminal and go to the cloned repository's directory
3. Run the command - '*npm install'* to install the node dependencies
4. Install elastic search referring to these links - Windows/MacOS/Linux / MacOS

Start the Stories microservice:

1. Open a terminal to run the local database. Run the command '*mongo'* to start the mongo database
2. Open the cloned repository in another terminal. Run the command '*npm start'* to start the the Stories microservice
3. Open Postman or the web browser to start using and running the API endpoints

Folder structure

i) **dao** - consists of all the database level interactions
   (a) comment.dao.server.js
   (b) es.story.dao.server.js
   (c) media.dao.server.js
   (d) taxonomy.dao.server.js

ii) **db** - Handles connection to the database
   (i) db.js
   (ii) elasticsearch.js

iii) **models** - Creates a database model for all the schemas used in Stories microservice
   (a) comments.model.server.js
   (b) media.model.server.js
   (c) story.model.server.js
   (d) taxonomy.model.server.js

iv) **routes** - Consists of routing logic from the database level to the API level
   (a) es.story.route.server.js

v) **schemas** - Consists of all the database schemas for the microservice
   (a) comment.schema.server.js
   (b) media.schema.server.js
   (c) story.schema.server.js
   (d) taxonomy.schema.server.js

vi) **tests** - Consists of all the unit test for the microservice
   (a) comment.test.js
   (b) db.handler.js
   (c) story.test.js
   (d) taxonomy.test.js

Database structure

Story schema:

Column	Datatype
story_id (indexed)	String
posted_by	String
user_id (indexed)	Number
Hyperlink	String
rating (indexed)	Number
story_title (indexed)	String
place_ids (indexed)	Array(Number)
media_type (indexed)	String
date (indexed)	Date
solution (indexed)	Array(String)
sector (indexed)	Array(String)
description (indexed)	String
strategy (indexed)	Array(String)
Comments	Array(CommentSchema)
liked_by_users	Array(Number)
flagged_by_users (indexed)	Array(Number)

Comment Schema:

Column	Datatype
comment_id	String
user_id	Number
user_name	String
Content	String
Date	Date

Taxonomy schema:

Column	Datatype
Strategy	String
Sector	String
Solution	String

Mediatype schema:

Column	Datatype
mediaType	String

APIs available

Stories API:

1. findAllStories
2. findStoryByStoryID
3. findStoryByPlaceID
4. findStoryByTitle
5. findTopStories
6. findUnratedStories
7. findStoryByDescription
8. findStoryByDescription
9. findStoryBySector
10. findStoryBySolution
11. findStoryByStrategy
12. getSortedByFlagged
13. findStoryByUserID
14. advancedSearch
15. createStory
16. deleteStory
17. updateStory
18. likeStory
19. unlikeStory
20. flagStory
21. unflagStory
22. addRatingToStory

Comments API:

1. addComment
2. findAllComments
3. deleteComment

Story Preview API:

1. getPreview

Taxonomy API:

1. findAllTaxonomy
2. findTaxonomyBySolution
3. findTaxonomyByStrategy
4. findTaxonomyBySector
5. findAllSolution
6. findAllSector

MediaType API:

1. getAllMediaTypes

Known Issues and Risks

Issues reported for the Stories microserver are posted here

Future Features

1. Add emoticons in addition to likes for stories
2. Add additional sorting filters to stories