Blazing Point Articles
Interview Project Documentation

Garrett Christian

Project Description

In November of 2019 I applied for internship at Perfect Sense. For the technical evaluation I was tasked with:
"Using any language or framework, we ask that you design and build an article or blog post on which others can submit comments. Your code should touch on both the front end and the back end."
I created and presented Perfect Articles and was offered an internship position which unfortunately fell though due to Coronavirus. Then in October following a career fair at JMU I was tasked with the same evaluation project by Perfect Sense, now Bright spot. So, in preparation for the interview I have rebranded my project as Blazing Point Articles.

Contents

Change Log

Revamped: 10/4/20

- Changed the name to Blazing Point Articles
- Created a new logo
- Added 3 new articles
- Made the documentation more robust

Created: 12/18/19

- Created the front-end and back end of Perfect Articles

Implementation Details

Process (12/18/19)

- Began working on the project in the front end since that's where I had less experience.
- Used https://www.springboottutorial.com/spring-boot-react-full-stack-crud-mavenapplication as a guide.
- Determined the look of the components first, filling in with dummy data in the front end.
- Added the Axios calls to get data from the backend.
- Created the endpoints in the backend to give dummy data.
- Integrated that new data into the front end.
- Built out the backend to persist data to the database.
- Tested functionality through postman and the front end.
- Finalized the documentation and cleaned the code.

Technology Stack

Built the project from these starting points:

Front end	Back end	Database
• npx create-react-app	• Spring Initializr	• Docker PostgreSQL container

Applications used:

Front end	Back end	Database
• Visual Studio Code	• IntelliJ • Postman	• Kitematic • DataGrip

Languages / Frameworks used to create the project:

Front end	Back end	Database
• Java Script • React • CSS • html • Axios • Bootstrap	• Java • Spring • Hibernate • Maven • Json	• Sql • Postres

Steps to start the project locally

1. Launch the Postgres database docker container.
2. If necessary, correct the data source url with the new port number the container is running on, in: perfectArticlesBack/src/main/resources/application.properties. The property is: spring.datasource.url= jdbc:postgresql://localhost:32768/postgres.
3. Launch perfectArticlesBack.
4. In the terminal, change directories into perfect-articles-frontend and run: npm start.
5. Click the BPA logo to the left of the title on the to populate the database with example articles (refresh).

Test articles from:

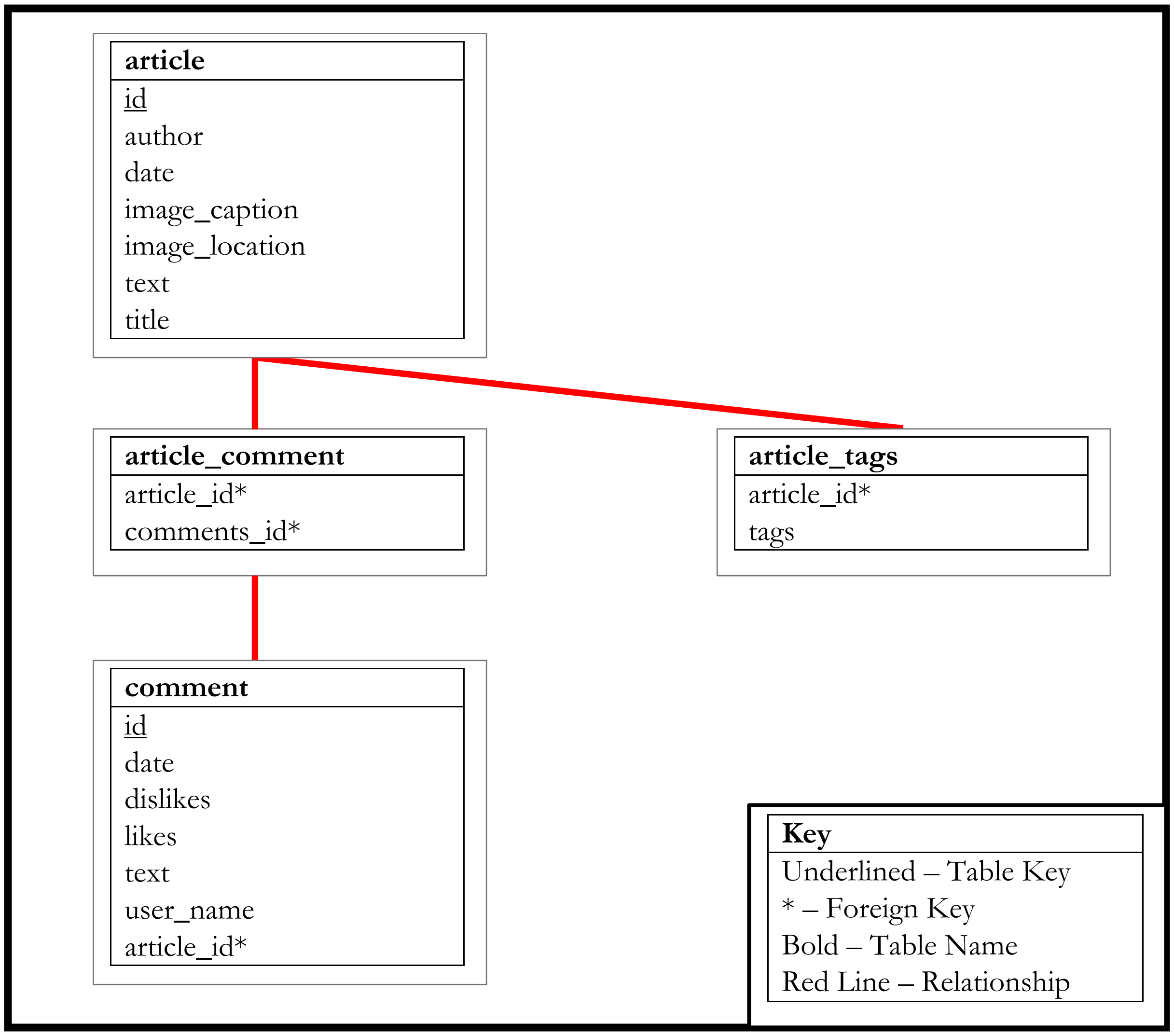
- https://en.wikipedia.org/wiki/Strawberry
- https://thethoughtfulgamer.com/2017/06/02/viticulture-essential-edition-review/
- https://thirstkey.com/kota-the-friend-everything-album-review/#:~:text=%E2%80%9CEverything%E2%80%9D%20is%20a%20very%20necessary,minutes%2C%20but%20a%20worthwhile%20one.
- https://www.newyorker.com/books/page-turner/the-difference-between-bird-watchingand-birding
- https://pitchfork.com/reviews/albums/harry-styles-fine-line/
- https://www.shutupandsitdown.com/review-sagrada/

Database Entity Relationship Diagram

Tables

article	Table of articles
article_comment	Join table for the articles and comments joined by their respective IDs
article_tags	Table of tags for articles joined by the articles ID
comment	Table of comments owned by articles

Diagram



Endpoints

ArticleController:

getAllArticles:

Path	http://localhost:8080/article/full
Type	Get
Description	Gives you all articles ordered by the most recently posted article
Return Type	List<ArticleDto>

getArticle:

Path	http://localhost:8080/article/{id}/id
Type	Get
PathVariable	{id} – Integer and must map to a valid article
Description	Gives you the articleDto of the the {id} passed in if valid if it is not valid this method will return null
Return Type	ArticleDto

getArticlesByDate:

Path	http://localhost:8080/article/{amount}/amount
Type	Get
PathVariable	{amount} – Integer
Description	Gives you the {amount} of articleDtos requested ordered by most recently posted article
Return Type	List<ArticleDto>

loadExampleArticles:

Path	http://localhost:8080/article/articles
Type	Post
Description	Adds the example articles to the database
Return Type	AddResponse

CommentController:

getArticleComments:

Path	http://localhost:8080/comment/{idArticle}/id_article
Type	Get
PathVariable	{idArticle} – Integer must map to valid article
Description	Gives you all the comments for the requested {idArticle}
Return Type	List<CommentDto>

getArticleCommentTop

Path	http://localhost:8080/comment/top/{idArticle}/id_article
Type	Get
PathVariable	{idArticle} – Integer must map to valid article
Description	Gives you the most recent comment on an article
Return Type	CommentDto

getCommentById:

Path	http://localhost:8080/comment/{id}/id
Type	Get
PathVariable	{is} – Integer must map to valid comment
Description	Gives you the comment with {id}
Return Type	CommentDto

addCommentToArticle:

Path	http://localhost:8080/comment/comment/{idArticle}/id_article
Type	Post
PathVariable	{idArticle} – Integer must map to valid article
RequestBody	CommentDto
RequestBody example	{ "userName": "Garrett Christian", "text": "This was a great read!", "likes": 0, "dislikes": 0 }
Description	Adds the RequestBody comment to the article date and id will be ignored the username and text must be at least one character
Return Type	AddResponse

changeCommentLikeDislikes:

Path	http://localhost:8080/comment/{id}/id/{likeDislikes}/like_dislikes/{value}/value
Type	Post
PathVariable	{id} – Integer must map to a valid comment id {likeDislikes} – String use “LIKES” to change likes and “DISLIKES” to change dislikes {value} – Integer how much will be add to the current value of likes/dislikes
Description	Updates the comment’s like/dislike by the amount of {value}
Return Type	AddResponse

Backlog for Future Sprints

Backlog

Feature Add Users	Add Users – add users to the perfect article project Requirements: • Add new user classes to the backend with a one to many relationships with comments • Create new endpoints in the backend to support the user class • Implement users in the front end with login page and automatic usernames for comments
Improvement Pageless pagination	Pageless pagination – Change the load all buttons to use pageless batching Requirements: • Change the current get all comments/article endpoints to accept an amount and sort by parameter • Add buttons to front end to support a batched list of comments/articles