API SUMMARY

API METHODS - DEFAULT

deleteFavourites
getArticles
getArticlesTitle
getFavourites
getUser
postFavourite

OPSC7312 Backend API docs

API and SDK Documentation

Version: 1.0.0

Introduction

This API is designed to support a simple Android application that allows users to view, search, and save news articles to their favorites, as well as access their account information.

It utilizes Hono.js, a lightweight and high-performance web framework that is faster than Express.js, making it ideal for speed and efficiency. The API is hosted on Vercel, taking advantage of its ability to run on the Edge, which ensures ultra-low latency by serving requests closer to the user's location.

By leveraging this tech stack, the API provides a responsive and scalable backend, ensuring fast response times and a reliable experience for the end user.

Where the News is Sourced

The API retrieves news articles from public RSS feeds provided by various news publishers, focusing on South African news. Currently, the API sources articles from News24, with plans to add more sources.

- **Data Source:** The API queries RSS feed URLs, which provide news content in XML format.
- **RSS Feed Description:** An RSS feed (Really Simple Syndication) is a standardized XML format for delivering regularly updated web content, such as news articles or blog posts.
- **Parsing:** Using Cheerio, the API parses the XML data to extract relevant information, such as article titles, content, publication dates, and authors.

Delivery: The extracted data is formatted into structured JSON responses and sent to the client.

Evaluation this approach

Pros:

- Efficiency: Utilizes readily available RSS feeds with minimal setup.
- Flexibility: Cheerio allows for adaptation to various feed structures.
- Scalability: Easy to integrate additional sources. -Focus: Relevant South African news enhances user engagement.

Cons:

- Data Quality: Inconsistent formats and quality from third-party feeds.
- Dependency: Reliant on external feed reliability and availability.
- Control: Limited control over content and update frequency.
- Complexity: Managing multiple feeds can become complex.

Tech stack

- Hono.js: Web framework for the API
- Node.js: Runtime environment
- Vercel: Hosting platform
- Zod: Schema validation for type safety
- Cheerio: RSS feed parsing
- Jest: Testing framework
- Firebase Auth: Authentication service
- Firestore: NoSQL database for data storage

How it consumes and sends data

The API uses the standard **HTTP protocol** for simplicity and compatibility. It receives requests and payloads through the request URL and request body, respectively. After processing these requests, the API fulfills its operations and returns a structured response in **JSON format**.

- **Requests:** The API receives data in predefined structures, including parameters in the request URL (e.g., article IDs, user IDs) and payloads in the request body (e.g., user actions like adding an article to favorites).
- **Validation: Zod schemas** are employed within the API to validate incoming and outgoing objects, ensuring strict type safety. This validation ensures that all data adheres to the expected structure and types, reducing the risk of errors and improving overall data integrity.
- **Responses:** The API returns structured JSON objects, containing the requested data or the outcome of an operation (e.g., a list of articles, confirmation messages).

The data structures for both requests and responses are predefined, validated using Zod schemas, and agreed upon between the front-end and backend teams. This approach ensures consistency, strict type safety, and efficient data handling across the application.

Architecture and Design

The API is designed to provide the necessary CRUD operations for the Android application, with a focus on clean architecture and solid design principles.

- **Endpoint Operations:** Each API endpoint is responsible for handling specific CRUD operations required by the front-end, such as fetching articles, managing favorites, and accessing user information.
- **Backend Authentication:** The API uses backend authentication to validate incoming requests, ensuring that only authorized users can access or modify data.
- **Business Service Layer:** After authentication, data payloads are processed by a dedicated business service layer. This layer handles the core functionality and business logic, such as interacting with Firebase or implementing specific operations.
- **Test-Driven Development (TDD):** A test-driven development approach will be adopted to ensure synchronous development between the backend and frontend teams. This ensures that all features are thoroughly tested, reducing bugs and improving collaboration.
- **Rate Limiting:** Rate limiting features will be implemented to prevent abuse of the API, ensuring that requests are managed effectively and that the service remains available and secure for all users.

This architecture ensures a clear separation of concerns, with endpoints managing request handling and authentication, while the business service layer focuses on the implementation of operations. This design promotes maintainability, scalability, and a solid software architecture.

Hosting

The API will be hosted on Vercel, leveraging its edge network capabilities to ensure ultra-low latency and fast response times. This platform is chosen for its simplicity and efficiency, offering seamless integration with our tech stack while minimizing deployment complexity.

Simplicity: Vercel provides a streamlined deployment process, allowing us to focus on development rather than infrastructure management.

Efficiency: The API utilizes Firebase services for authentication and NoSQL database storage, ensuring secure and efficient data management. Combined with Vercel's

edge network, the API serves requests from locations closest to users, optimizing performance and reducing load times.

Minimal Cost: Vercel's cost-effective pricing model, alongside Firebase's scalable services, keeps operational expenses low, ensuring the API remains affordable while delivering high-quality service.

This hosting strategy ensures that the API is efficient, secure, and cost-effective, providing a reliable backend solution with minimal overhead.

Default

deleteFavourites

Your DELETE endpoint

Remove an article from the user's favorites.

DELETE

```
/users/{public_id}/favourites
```

Usage and SDK Samples

Curl | Java | Android | Obj-C | JavaScript | C# | PHP | Perl | Python

```
curl -X DELETE\
"https://www.opsc7312.nerfdesigns.com/users/{public_id}/favourites"
```

Parameters

Path parameters

Name	Description
public_id*	String Required

Responses

Status: 500 - Internal Server Error, Couldn't remove article from favourites.

Status: 2XX - Confirmation message that the article has been removed from favorites.

getArticles

Your GET endpoint

Retrieve a list of news articles.

GET

```
/articles
```

Usage and SDK Samples

Curl | Java | Android | Obj-C | JavaScript | C# | PHP | Perl | Python

```
curl -X GET\
"https://www.opsc7312.nerfdesigns.com/articles?category=&limit=&offset="
```

Parameters

Query parameters

Name	Description
category	String *Filter articles by category.*
limit	String *Number of articles to retrieve.*
offset	String *Pagination offset.*

Responses

Status: 2XX - A list of articles with details such as title, content, category, publication date, image URL, and author.

getArticlesTitle

Your GET endpoint

Retrieve a specific news article by its ID.

GET

```
/articles/{title}
```

Usage and SDK Samples

Curl Java Android Obj-C JavaScript C# PHP Perl Python

```
curl -X GET\
"https://www.opsc7312.nerfdesigns.com/articles/{title}"
```

Parameters

Path parameters

Name	Description
title*	String Required

Responses

Status: 500 - Internal Server Error, couldn't get specific article.

Status: 2XX - The full details of the requested article.

getFavourites

Your GET endpoint

Retrieve the user's list of favorite articles.

GET

```
/users/{public_id}/favourites
```

Usage and SDK Samples

Curl Java Android Obj-C JavaScript C# PHP Perl Python

```
curl -X GET\
"https://www.opsc7312.nerfdesigns.com/users/{public_id}/favourites"
```

Parameters

Path parameters

Name	Description
public_id*	String Required

Responses

Status: 200 - A list of articles that the user has marked as favorites.

Status: 500 - Internal Server Error, Couldn't retrieve favourites.

getUser

Your GET endpoint

Retrieve the user's account information.

GET

```
/users/{public_id}
```

Usage and SDK Samples

Curl Java Android Obj-C JavaScript C# PHP Perl Python

```
curl -X GET\
"https://www.opsc7312.nerfdesigns.com/users/{public_id}"
```

Parameters

Path parameters

Name	Description
public_id*	String Required

Responses

Status: 500 - Internal Server Error, couldn't get user's info.

Status: 2XX - The user's account details, including name, email, and a list of favorite article IDs.

postFavourite

Your POST endpoint

Add an article to the user's favorites.

POST

/users/{public_id}/favourites

Usage and SDK Samples

Curl | Java | Android | Obj-C | JavaScript | C# | PHP | Perl | Python

```
curl -X POST\
"https://www.opsc7312.nerfdesigns.com/users/{public_id}/favourites"
```

Parameters

Path parameters

Name	Description
public_id*	String Required

Responses

Status: 500 - Internal Server Error, Couldn't add article to favourites.

Status: 2XX - A list of articles that the user has marked as favorites.

Suggestions, contact, support and error reporting;

Information URL: nerfdesigns.com
Contact Info: charlierossouw@outlook.com