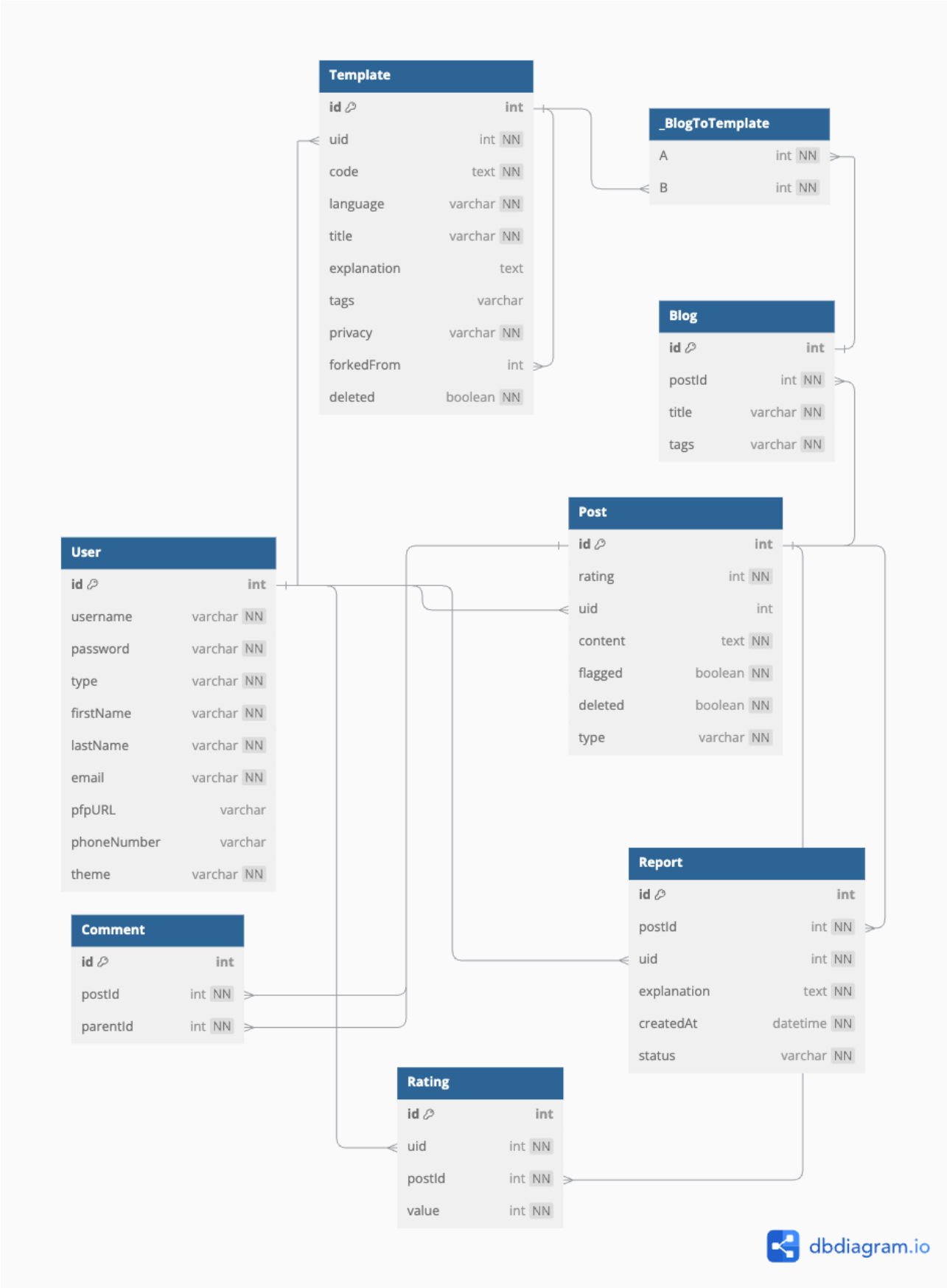
ADMIN Credentials:

- Username: admin
- Password: admin

**For possible convenience running our postman scripts, we were testing our API on postman with local environment variables {{auth}} and {{refreshAuth}}*



Model Design

API Documentation

Scriptorium

November 5, 2024

Contents

1 User Registration Endpoint

This endpoint allows new users to register by providing essential account details.

Endpoint

`/api/users/register`

Method

POST

Payload

- `username` (string, required) - Unique username for the user.
- `password` (string, required) - Password for the user account.
- `type` (string, optional) - Account type (default: "USER").
- `firstName` (string, required) - First name of the user.
- `lastName` (string, required) - Last name of the user.
- `email` (string, required) - Unique email for the user.
- `pfpURL` (string, optional) - URL for profile picture.
- `phoneNumber` (string, optional) - User's phone number.
- `theme` (string, optional) - Preferred theme setting.

Example Request

```
{
  "username": "exampleUser",
  "password": "securePassword123",
  "type": "USER",
  "firstName": "John",
  "lastName": "Doe",
  "email": "john.doe@example.com",
  "pfpURL": "https://example.com/profile.jpg",
  "phoneNumber": "123-456-7890",
  "theme": "LIGHT"
}
```

Responses

201 Created

```
{
  "message": "User registered successfully.",
  "user": {
    "id": 1,
    "username": "exampleUser"
  }
}
```

2 User Login Endpoint

This endpoint allows users to authenticate by providing their username and password. Upon successful authentication, it returns an access and refresh token.

Endpoint

`/api/users/login`

Method

POST

Payload

- `username` (string, required) - Unique username of the user.
- `password` (string, required) - Password for the user account.

Example Request

```
{
  "username": "exampleUser",
  "password": "securePassword123"
}
```

Response

200 OK

```
{
  "accessToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
  "refreshToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9..."
}
```

3 Refresh Access Token Endpoint

This endpoint generates a new access token when provided with a valid refresh token, allowing users to maintain authenticated sessions.

Endpoint

`/api/users/refresh`

Method

POST

Payload

- `refreshToken` (string, required) - A valid refresh token used to request a new access token.

Example Request

```
{
  "refreshToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9..."
}
```

Response

200 OK

```
{
  "accessToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9..."
}
```

4 Update User Profile Endpoint

This endpoint allows an authenticated user to update their profile information, including name, email, profile picture URL, phone number, and theme.

Endpoint

`/api/users/profile`

Method

PUT

Payload

- `username` (string, required) - The username of the user whose profile is to be updated.
- `firstName` (string, required) - User's first name.
- `lastName` (string, required) - User's last name.
- `email` (string, required) - User's email address. Must be unique.
- `pfpURL` (string, optional) - URL to the user's profile image (must be from predefined options).
- `phoneNumber` (string, optional) - User's phone number.
- `theme` (string, optional) - Preferred theme (e.g., LIGHT or DARK).

Example Request

```
{
  "username": "john_doe",
  "firstName": "John",
  "lastName": "Doe",
  "email": "john.doe@example.com",
  "pfpURL": "https://example.com/profile.jpg",
  "phoneNumber": "+1234567890",
  "theme": "DARK"
}
```

Response

200 OK

```
{
  "id": 1,
  "username": "john_doe",
  "firstName": "John",
  "lastName": "Doe",
  "email": "john.doe@example.com",
  "pfpURL": "https://example.com/profile.jpg",
  "phoneNumber": "+1234567890",
  "theme": "DARK"
}
```

5 Code Execution Endpoint

This endpoint allows a user to submit code in various programming languages for execution. The server compiles or interprets the code and returns the output. Supports C, C++, Java, Python, and JavaScript.

Endpoint

`/api/execute`

Method

POST

Payload

- `code` (string, required) - The code to be executed.
- `language` (string, required) - Language of the code. Supported values: python, javascript, c, cpp, java.
- `stdin` (string, optional) - Input to pass to the program. Newlines can be specified with `\n`.

Example Request

```
{
  "code": "print('Hello, World!')",
  "language": "python",
  "stdin": ""
}
```

Response

200 OK
If the code executes successfully:

```
{
  "output": "Hello, World!\n",
  "error": ""
}
```

If there is a compilation or runtime error: **500 Internal Server Error**

```
{
  "error": "Execution error",
  "details": "Traceback (most recent call last): ..."
}
```

6 Template Creation Endpoint

This endpoint allows an authenticated user to create a new code template.

Endpoint

`/api/templates/users/create`

Method

POST

Payload

- `title` (string, required) - The title of the template (max: 100 characters).
- `explanation` (string, required) - A detailed explanation of the template's purpose.
- `tags` (string, optional) - Tags in CSV format without spaces to categorize the template.
- `code` (string, required) - The code for the template.
- `language` (string, required) - Programming language of the code (e.g., "JavaScript", "Python").

Example Request

```
{
  "title": "Sample Template",
  "explanation": "This template demonstrates a basic code
    ↪ structure.",
  "tags": "javascript,basics",
  "code": "console.log('Hello, World!');",
  "language": "javascript"
}
```

Response

200 OK
If the template is created successfully:

```
{
  "id": 123,
  "message": "Successfully created new template"
}
```

7 Template Forking Endpoint

This endpoint allows an authenticated user to fork (duplicate) a public template, creating a new template in the user's account based on the specified template ID.

Endpoint

`/api/templates/users/create-fork`

Method

POST

Payload

- `id` (integer, required) - The ID of the public template to be forked.

Example Request

```
{
  "id": 123
}
```

Response

200 OK
If the template is forked successfully:

```
{
  "id" : "240",
  "message": "Successfully forked template"
}
```

8 Template Update and Delete Endpoint

This endpoint allows an authenticated user to update or delete their own template based on the provided template ID. Update requests modify specified fields, while delete requests mark the template as deleted.

Endpoint

`/api/templates/users/edit`

Method

PUT - Update template
DELETE - Delete template

Payload (PUT only)

- `id` (integer, required) - The ID of the template to be updated or deleted.
- `title` (string, optional) - The new title for the template.
- `explanation` (string, optional) - The new explanation for the template.
- `tags` (string, optional) - A CSV string of tags.
- `code` (string, optional) - The new code content.
- `language` (string, optional) - The programming language of the template.
- `privacy` (string, optional) - Privacy setting, must be one of `PUBLIC` or `PRIVATE`.

Example Request - Update (PUT)

```
{
  "id": 123,
  "title": "New Template Title",
  "explanation": "Updated explanation",
  "tags": "example,template",
  "code": "print('Hello, World!')",
  "language": "python",
  "privacy": "PUBLIC"
}
```

Example Request - Delete (DELETE)

```
{
  "id": 123
}
```

Response

200 OK
For successful update:

```
{
  "message": "Successfully updated template"
}
```

For successful deletion:

```
{
  "message": "Successfully deleted template"
}
```

9 Get All User Templates

This endpoint retrieves all templates created by an authenticated user. It returns each template's ID, title, and tags.

Endpoint

`/api/templates/users/all`

Method

GET - Retrieve all templates for the authenticated user

Payload

None

Example Request

```
GET /api/templates/user/all
Headers: {
  "Authorization": "Bearer <access_token>"
}
```

Response

200 OK
A list of templates with each template's ID, title, and tags:

```
[
  {
    "id": 101,
    "title": "Template One",
    "tags": "example,template"
  },
  {
    "id": 102,
    "title": "Template Two",
    "tags": "sample,code"
  }
]
```

10 Get Specific Public Template

This endpoint retrieves a specific template if it is set to `PUBLIC` visibility.

Endpoint

`/api/templates/[id]`

Method

GET - Retrieve a specific public template by ID

Query Parameters

- `id` - The unique identifier of the template to retrieve (integer)

Example Request

```
GET /api/templates/123
```

Response

200 OK
Returns the requested template data if it is public:

```
{
  "id": 123,
  "uid": 42,
  "title": "My Template",
  "code": "print('Hello, world!')",
  "language": "python",
  "tags": "example,template",
  "privacy": "PUBLIC",
  "deleted": false
}
```

11 Get Public Templates

This endpoint retrieves a list of public templates based on specified search criteria.

Endpoint

`/api/templates/`

Method

GET - Retrieve a list of public templates with optional filtering

Query Parameters

- `skip` (optional) - The number of records to skip (for pagination).
- `take` (optional) - The number of records to retrieve (for pagination).
- `title` (optional) - A substring to match in the template title.
- `tags` (optional) - A comma-separated string of tags to match (no spaces).
- `content` (optional) - A substring to match in the template explanation.

Example Request

```
GET /api/templates/search?skip=0&take=10&title=Example&tags=
   ↪ javascript,node
Headers: {
  "Authorization": "Bearer <access_token>"
}
```

Response

200 OK
Returns an array of public templates that match the search criteria:

```
{
  "data": [
    {
      "id": 123,
      "title": "Example Template",
      "explanation": "A sample template for demonstration",
      "tags": "javascript,node"
    },
    {
      "id": 124,
      "title": "Another Template",
      "explanation": "Another example template",
      "tags": "typescript,express"
    }
  ],
  "message": null,
  "isEmpty": false
}
```

400 Bad Request
If tags are not provided in the correct CSV format or no templates are found:

```
{
  "data": [],
  "message": "No templates were found. Try loosening your search
    ↪ and check spelling.",
  "isEmpty": true
}
```

12 Create Blog Post

This endpoint allows an authenticated user to create a new blog post with optional tags and linked templates.

Endpoint

`/api/posts/blogs/users/create`

Method

POST - Create a new blog post with specified content, title, tags, and templates.

Payload

- `title` (required, string) - The title of the blog post (max 100 characters).

- `description` (required, string) - The main content of the blog post (max 5000 characters).
- `tags` (optional, string) - Comma-separated tags for the blog post, formatted as CSV without spaces (max 100 characters).
- `templates` (optional, array of integers) - Array of template IDs linked to this blog.

Example Request

```
POST /api/posts/blogs/create
Headers: {
  "Authorization": "Bearer <access_token>"
}
Body: {
  "title": "How to Use Prisma",
  "description": "A guide on how to use Prisma ORM effectively in
    ↪ web applications.",
  "tags": "prisma,orm,javascript",
  "templates": [1, 2, 3]
}
```

Response

200 OK
Successfully created the blog post:

```
{
  "id": 101,
  "message": "Successfully created new blog"
}
```

13 Update/Delete Blog Post

This endpoint allows an authenticated user to update or delete a specified blog post.

Endpoint

`/api/posts/blogs/users/edit`

Methods

- **PUT** - Updates the blog post title, description, tags, and templates.
- **DELETE** - Deletes the specified blog post by marking it as deleted and clearing its content.

Payload (PUT only)

- `id` (required, integer) - ID of the blog post.
- `title` (optional, string) - Title of the blog post.
- `description` (optional, string) - Main content of the blog post.
- `tags` (optional, string) - Comma-separated tags for the blog post, formatted as CSV without spaces.
- `templates` (optional, array of integers) - Array of template IDs linked to this blog.

Example Request (PUT)

```
PUT /api/posts/blogs/edit
Headers: {
  "Authorization": "Bearer <access_token>"
}
Body: {
  "title": "Updated Blog Title",
  "description": "Updated blog content.",
  "tags": "updated,blog,post",
  "templates": [1, 2]
}
```

Example Request (DELETE)

```
DELETE /api/posts/blogs/edit
Headers: {
  "Authorization": "Bearer <access_token>"
}
Body: {
  "id": 123
}
```

Response

200 OK

If the update or deletion was successful:

```
{
  "message": "Successfully updated blog"
}
// or
{
  "message": "Successfully deleted blog"
}
```

14 Get Blog Post by ID

This endpoint allows a user to retrieve a specific blog post by its ID.

Endpoint

`/api/posts/blogs/[id]`

Method

GET - Retrieves a specific blog post's data.

Query Parameters

- `id` (required, integer) - The unique ID of the blog post.

Example Request

```
GET /api/posts/blogs/123
```

Response

200 OK
If the blog post retrieval was successful:

```
{
  "id": 123,
  "rating": 5,
  "uid": 1,
  "replies": [],
  "content": "This is the blog content.",
  "flagged": false,
  "deleted": false,
  "title": "Blog Title",
  "tags": "tag1,tag2,tag3"
}
```

15 Get Filtered Blog List

This endpoint allows a user to retrieve a paginated and filtered list of blog posts based on specified criteria such as title, tags, content, and template title. The list is ordered by blog rating in descending order.

Endpoint

`/api/posts/blogs`

Method

GET - Retrieves a filtered and paginated list of blog posts.

Query Parameters

- `skip` (optional, integer) - Number of results to skip.
- `take` (optional, integer) - Number of results to return.
- `blogTitle` (optional, string) - Filter by blog title.
- `desiredContent` (optional, string) - Filter by content within the blog post.
- `blogTags` (optional, string) - Filter by CSV-formatted tags.
- `templateTitle` (optional, string) - Filter by template title associated with the blog.

Example Request

```
GET /api/posts/blogs/list?skip=0&take=10&blogTitle=JavaScript&
   ↪ desiredContent=tutorial&blogTags=frontend&templateTitle=
   ↪ javascript
```

Response

200 OK

If the blog posts retrieval was successful:

```
{
  "data": [
    {
      "postId": 1,
      "title": "Intro to JavaScript",
      "post": {
        "id": 1,
        "rating": 5,
        "content": "JavaScript basics",
        "flagged": false,
        "deleted": false
      },
      "templates": [
        {
          "id": 1,
          "title": "JavaScript Template",
          "privacy": "PUBLIC"
        }
      ]
    }
  ],
  "message": null,
  "isEmpty": false
}
```

400 Bad Request
If an invalid value is given for tags or no blogs match the filters:

```
{
  "data": [],
  "message": "No blog was found. Try loosening your search and
    ↪ check spelling.",
  "isEmpty": true
}
```

16 Create Comment

This endpoint allows an authenticated user to create a comment on a specific post.

Endpoint

`/api/posts/comments/create`

Method

POST - Creates a comment under a specified post.

Payload

- `id` (integer, required) - The ID of the post being commented on.
- `description` (string, required) - The content of the comment, with a maximum length of 1000 characters.

Example Request

```
POST /api/posts/comments/create
Headers: {
  "Authorization": "Bearer <access_token>"
}
{
  "id": 123,
  "description": "This is my comment on the post."
}
```

Response

200 OK
If the comment is created successfully:

```
{
  "id" : 125,
  "message": "Successfully created comment"
}
```

17 Get Post Replies

This endpoint retrieves a paginated list of replies to a specific post, ordered by rating.

Endpoint

`/api/posts/shared/replies`

Method

GET - Retrieves replies to a specific post.

Payload

- `id` (integer, required) - The ID of the parent post to retrieve replies for.
- `skip` (integer, optional) - The number of replies to skip (for pagination).
- `take` (integer, optional) - The number of replies to retrieve (for pagination).

Example Request

```
GET /api/posts/comments/replies
{
  "id": 123,
  "skip": 0,
  "take": 10
}
```

Response

200 OK

If replies are successfully retrieved:

```
{
  "data": [
    {
      "id": 1,
      "rating": 5,
      "content": "This is a reply to the post."
    },
    {
      "id": 2,
      "rating": 3,
      "content": "Another reply to the post."
    }
  ],
  "message": null
}
```

18 Update Post Vote

This endpoint allows a user to vote (upvote, downvote, or reset) on a specific post. The vote is saved to the user's rating on the post, and the post's cumulative rating is updated accordingly.

Endpoint

`/api/posts/shared/vote`

Method

PUT - Updates the user's vote on a specific post.

Payload

- `id` (integer, required) - The ID of the post to vote on.
- `rating` (integer, required) - The vote value, where:
  - `1` = Upvote
  - `-1` = Downvote
  - `0` = Reset vote (remove upvote or downvote)

Example Request

```
PUT /api/posts/vote
Headers: {
  "Authorization": "Bearer <access_token>"
}
{
  "id": 123,
  "rating": 1
}
```

Response

200 OK
If the vote is successfully recorded:

```
{
  "message": "Successfully updated user vote on the post"
}
```