The purpose of this document is solely to describe the programmed API, showing how to proceed with each endpoint, with information to send, what to expect.

This collection is divided in two major parts: the first one is the public API, where any one can use it, and the second one is the private API, where only authorized people can use it.

Colors were used to show easily expected behavior on normal flow (green), mistakes caused by the user (yellow) and mistakes caused by the system (red)

**PUBLIC API**

URI: /api/login

METHOD: POST

Keys and values: valid email and a password between 3 and 16 characters long

Although it is not inserting new information, due to the information sensitivity, this is a POST method

TOKEN: If there is an account with the given email and corresponding password, it returns a TOKEN to be used in further requests that demand authentication.

INVALID EMAIL: an email was out of the expected parameters, like ‘example@test.com’. Not all accepted emails will be valid, but the algorithm filters the most part

PASSWORD MUST BE…: this message is sent if the password is not between 3 and 16 characters long

EMAIL AND/OR PASSWORD INCORRECT: if email and password are within the expected, but no match was found, return this message so it difficult the hack of the system

URI: /api/sign-up

METHOD: POST

Keys and values: valid email and a password between 3 and 16 characters long

USER SUCCESSFULLY REGISTERED: if all parameters are within expected, and there is no previous email registered, return this message

INVALID EMAIL: an email was out of the expected parameters, like ‘example@test.com’. Not all accepted emails will be valid, but the algorithm filters the most part

PASSWORD MUST BE…: this message is sent if the password is not between 3 and 16 characters long

EMAIL AND/OR PASSWORD INCORRECT: if email and password are within the expected, but no match was found, return this message so it difficult the hack of the system

EMAIL ALREADY REGISTERED: if email and password are within the expected, but a email match was found, return this message

URI: /api/articles

METHOD: GET

Keys and values: “category” as the key, and the value is the desired category. (ex.: it’s a API for articles about movies, so the categories might be ‘Terror’, ‘Police’, ‘Sci-Fi’)

result: if the parameter CATEGORY is set and there is a valid match with articles listed, it return all the articles within that category in the format below

[

  {

    "author": {

      "name": "Author Name",

      "picture": "https://picture.url"

    },

    "category": "Category",

    "title": "Article title",

    "summary": "This is a summary of the article"

  },

  ...

]

no result found: the parameter was set but no category was found or there are no articles in the listed category, returning the message “There are no Articles”. There is no error, message or warning if no category is sent, it is considered non existent, and return the same message

URI: /api/articles/:id

METHOD: GET

Keys and values: “id” as the key, and the value is the desired article (in the current development state, a numeric value). Usage example: the desired article has the ID of 5 => url.com/api/articles/5

result: if the parameter id is set and a match is found, it returns the desired article with the corresponding information as image below.

{

      "author": {

        "name": "Author Name",

        "picture": "https://picture.url"

      },

      "category": "Category",

      "title": "Article title",

      "summary": "This is a summary of the article",

      "firstParagraph": "<p>This is the first paragraph of this article</p>"

}

THERE IS NO ARTICLE WITH THE GIVEN ID: the parameter was set but no article found, returns the this message

**PRIVATE API**

URI: /api/admin/authors

METHOD: GET

Keys and values: ----

HEADER: a token must be passed to validate the user. This token is acquired in the public API, ‘/api/login’. It must be set as “KEY <> VALUE”, with the key ‘token’ and the value the actual token received

result: if user validated and there is at least one author in the DATABASE, return a list of all the authors in the system as below. The ID appears here so it can facilitate the information flow

{

      {

"id": 1,

        "name": "Author Name",

        "picture": "https://picture.url"

      },

. . .

}

NO AUTHOR WERE FOUND: if there are no authors in the DATABASE, this is the message returned

PERMISSION DENIED: if the token is not valid or there is not even a token to begin with, this message occurs.

URI: /api/admin/authors/:id

METHOD: GET

Keys and values: “id” as the key, and the value is the desired author. Usage example: the desired author has the ID of 5 => url.com/api/admin/authors/5 (passing a valid TOKEN)

HEADER: a token must be passed to validate the user. This token is acquired in the public API, ‘/api/login’. It must be set as “KEY <> VALUE”, with the key ‘token’ and the value the actual token received

result: if user validated and the desired author exists, return the information of the author in the system as below.

{

      {

"name": "Author Name",

        "picture": "https://picture.url"

      }

}

THERE IS NO AUTHOR WITH THE GIVEN ID: if user validated and no match was found, this message is delivered to the user

PERMISSION DENIED: if the token is not valid or there is not even a token to begin with, this message occurs.

URI: /api/admin/authors

METHOD: POST

Keys and values: demands two pair of key <> value. The first one is the Author Name (Key: name) and the value corresponding, and the second one is the URL for a profile picture (Key: picture) and the value the url.

HEADER: a token must be passed to validate the user. This token is acquired in the public API, ‘/api/login’. It must be set as “KEY <> VALUE”, with the key ‘token’ and the value the actual token received

AUTHOR SUCCESSFULLY REGISTERED: if user validated and there is no conflict with the given information (the author name and the profile picture must be unique, so the DATABASE does not have duplicates), the new author is inserted in the DB and this message returns

NAME AND/OR PICTURE URL ALREADY EXISTS: if user validated and there is a match with one (or both) information, this message is delivered to the user

MISSING PARAMETERS: if user validated but one or both values are not sent, this message occurs, since the DATABASE refuses insertion in this table with missing information, so none of the values can be null

PERMISSION DENIED: if the token is not valid or there is not even a token to begin with, this message occurs.

URI: /api/admin/authors/:id

METHOD: PUT

Keys and values: “id” as the key, and the value is the desired author. Usage example: the desired author has the ID of 5 => url.com/api/admin/authors/5 (passing a valid TOKEN). Besides this key, any information about the desired author to be changed must be passed forward (possible keys: ‘name’ and ‘picture’)

HEADER: a token must be passed to validate the user. This token is acquired in the public API, ‘/api/login’. It must be set as “KEY <> VALUE”, with the key ‘token’ and the value the actual token received

SUCCESSFUL UPDATE: if user validated, the desired author exists and the UPDATE occurs, return this message.

THERE IS A CONFLICT WITH ANOTHER AUTHOR: if user validated but there is a duplicated information, the UPDATE fails and this message is sent to the user

PERMISSION DENIED: if the token is not valid or there is not even a token to begin with, this message occurs.

URI: /api/admin/authors/:id

METHOD: DELETE (note: the deletion of an author, deletes all the articles of that author as well)

Keys and values: “id” as the key, and the value is the desired author. Usage example: the desired author has the ID of 5 => url.com/api/admin/authors/5 (passing a valid TOKEN).

HEADER: a token must be passed to validate the user. This token is acquired in the public API, ‘/api/login’. It must be set as “KEY <> VALUE”, with the key ‘token’ and the value the actual token received

SUCCESSFUL DELETION: if user validated, the desired author exists and the UPDATE occurs, return this message.

THERE IS NO AUTHOR WITH THE GIVEN ID: if user validated but there is no match with an author, this message is sent to the user

PERMISSION DENIED: if the token is not valid or there is not even a token to begin with, this message occurs.

URI: /api/admin/articles

METHOD: GET

Keys and values: ----

HEADER: a token must be passed to validate the user. This token is acquired in the public API, ‘/api/login’. It must be set as “KEY <> VALUE”, with the key ‘token’ and the value the actual token received

result: if user validated, returns, if any, of the articles in the system. The id comes along to make it easy to choose witch one to operate

[

  {

"id": 1,

    "author": {

      "name": "Author Name",

      "picture": "https://picture.url"

    },

    "category": "Category",

    "title": "Article title",

    "summary": "This is a summary of the article"

  },

  ...

]

NO ARTICLES WERE FOUND: if user validated but there is no article in the DATABASE, this message appears

PERMISSION DENIED: if the token is not valid or there is not even a token to begin with, this message occurs.

URI: /api/articles/:id \*\*Note that is the same URI as the public API URI. The only difference is the token passed, witch returns more information about the desired article

METHOD: GET

Keys and values: “id” as the key, and the value is the desired article (in the current development state, a numeric value). Usage example: the desired article has the ID of 5 => url.com/api/articles/5

HEADER: a token must be passed to validate the user. This token is acquired in the public API, ‘/api/login’. It must be set as “KEY <> VALUE”, with the key ‘token’ and the value the actual token received

result: if the parameter id is set , a match is found and it is a valid token, it returns the desired article with the corresponding information as image below. (Extra information: body)

{

      "author": {

        "name": "Author Name",

        "picture": "https://picture.url"

      },

      "category": "Category",

      "title": "Article title",

      "summary": "This is a summary of the article",

      "firstParagraph": "<p>This is the first paragraph of this article</p>",

      "body": "<div><p>Second paragraph</p><p>Third paragraph</p></div>"

}

THERE IS NO ARTICLE WITH THE GIVEN ID: the parameter was set but no article found, returns the this message

URI: /api/admin/articles

METHOD: POST

Keys and values: all the article information (title, summary, the first paragraph and the rest of the text) following the corresponding keys “title”, “summary”, “firstParagraph”, “body”, and two other as “authorId” and “category”, being the “authorId’ the only one to expect a numeric value.

HEADER: a token must be passed to validate the user. This token is acquired in the public API, ‘/api/login’. It must be set as “KEY <> VALUE”, with the key ‘token’ and the value the actual token received

SUCCESSFULLY REGISTERED: if user validated and all parameters are correct, inserts the new information to the DATABASE (if there is no category with the same description, a new one is created)

MISSING PARAMETERS: if user validated but there is some or all of the parameters missing, this message is presented

PERMISSION DENIED: if the token is not valid or there is not even a token to begin with, this message occurs.

URI: /api/admin/articles/:id

METHOD: PUT

Keys and values: one or more of the article information (title, summary, the first paragraph and the rest of the text) following the corresponding keys “title”, “summary”, “firstParagraph”, “body”, and two other as “authorId” and “category”, being the “authorId’ the only one to expect a numeric value. Also, the numeric value of “ID”, such as “url.com/api/admin/articles/5”

HEADER: a token must be passed to validate the user. This token is acquired in the public API, ‘/api/login’. It must be set as “KEY <> VALUE”, with the key ‘token’ and the value the actual token received

SUCCESSFULLY UPDATED: if user validated and all parameters are correct, updates all the given information

THERE IS NO ARTICLE TO UPDATE: if user validated but there is no article with the given ID, this message occurs

PERMISSION DENIED: if the token is not valid or there is not even a token to begin with, this message occurs.

URI: /api/admin/articles/:id

METHOD: DELETE

Keys and values: “id” as the key, and the value is the desired article (in the current development state, a numeric value). Usage example: the desired article has the ID of 5 => url.com/api/admin/articles/5

HEADER: a token must be passed to validate the user. This token is acquired in the public API, ‘/api/login’. It must be set as “KEY <> VALUE”, with the key ‘token’ and the value the actual token received

SUCCESSFULLY DELETED: if user validated and id of a valid article, deletes the information and returns this message

THERE IS NO ARTICLE TO DELETE: if user validated but there is no article with the given ID, this message occurs

PERMISSION DENIED: if the token is not valid or there is not even a token to begin with, this message occurs.