Catalog of a Library

Name: Andrés Felipe Sánchez Sánchez

Functional Requirements:

- System must show all the books of a given author to the customer.
- System must let the administrator register new books.
- System must let users (administrators and customers) register themselves.
- System must let customers and administrators login in the system with username and password.
- System must show all the books of a given category.
- System must show all the books by name (paginated).
- System must let the administrator delete the books.
- System must let the administrator update the data of the books.
- System must let the administrator create a new category of books.
- System must show all categories registered.
- System must let show authors by the name
- System must let the administrator register authors.
- System must show a customized list of books based on her/his last searches.

Non Functional Requirements:

- System should save the password of the customers and administrators encrypted.
- System should be available all time.
- System should be implemented as a restful API.
- System should encrypt all the traffic send to the restful API

Use Cases

Title	**Register User**
Primary Actor	Customer or Administrator
Success Scenario	The user (customer or Administrator) presses the button register and next, a form is shown to be filled. The form collects the following information: name, username, lastname, address, gender, phone, DNI and password. After filling the information, the user clicks the button “finish register”, then the system confirms that the DNI and username was not registered before and in case these conditions are met the register is completed successfully.

Title	**Login in the System**

Primary Actor	Customer or Administrator
Success Scenario	The user (customer or Administrator) presses the button login, then a form with username and password is shown. The user fills the fields of username and password and clicks the “submit” button. The system validates that the credentials are valid and sends a token to the user which must be sent after in every request, and also sends to the user his role in the system (Administrator or customer).

Title	**Show books by author (paginated)**
Primary Actor	Customer
Success Scenario	The customer types in the search input the name of an author and presses the button search, then the system returns the first page of books whose author contains the name. if the query clicks on the second page new authors are retrieved and so on.

Title	**Register a book**
Primary Actor	Administrator
Success Scenario	The customer presses the button “register new book”, then the system shows him a form with inputs to fill data like name, author, price, inventory, summary, category and isbn. When the administrator fills all data and presses the button “register” the system validates that there is not registered a book with the same isbn, Finally the system returns the book registered in the application.

Title	**Delete a book**

Primary Actor	Administrator
Success Scenario	When the administrator makes a query (by category, name or author) then the system shows the books and next to the name of each book the button “delete”, when the administrator clicks this button the book is deleted from the system.

Title	**Show books by category**
Primary Actor	Customer or Administrator
Success Scenario	The customer or Administrator types select in a dropdown a category and presses the button search, then the system returns the first page of books which belongs to the category specified. the user will be able to query for the next page of books.

Title	**Show books by name**
Primary Actor	Customer or Administrator
Success Scenario	The customer or Administrator types in the search input the name of a book and presses the button search, then the system returns the first page of books whose name contains the input.The user will be able to query for the next page of books.

Title	**Update data of a book**
Primary Actor	Administrator

Success Scenario	When the administrator makes a query (by category, name or author) then the system shows the books and next to the name of each book the button “edit”, when the administrator clicks this button the system shows to the user a form prefilled with the data of the book, the user then can edit each field and press the button “update”, after that the system update the book and returns the data of the book updated.

Title	**Register a Category of books**
Primary Actor	Administrator
Success Scenario	The administrator presses the button “add category” then a form with the inputs of name and description is shown to the user. When the user fills the data of the input fields and presses the button “register” the systems saves the category.

Title	**Books preferences for User**
Primary Actor	Customer or Administrator
Success Scenario	After successful login and multiple previous searches of the book. The next searches of books will list the books based on his/her preferences.

.

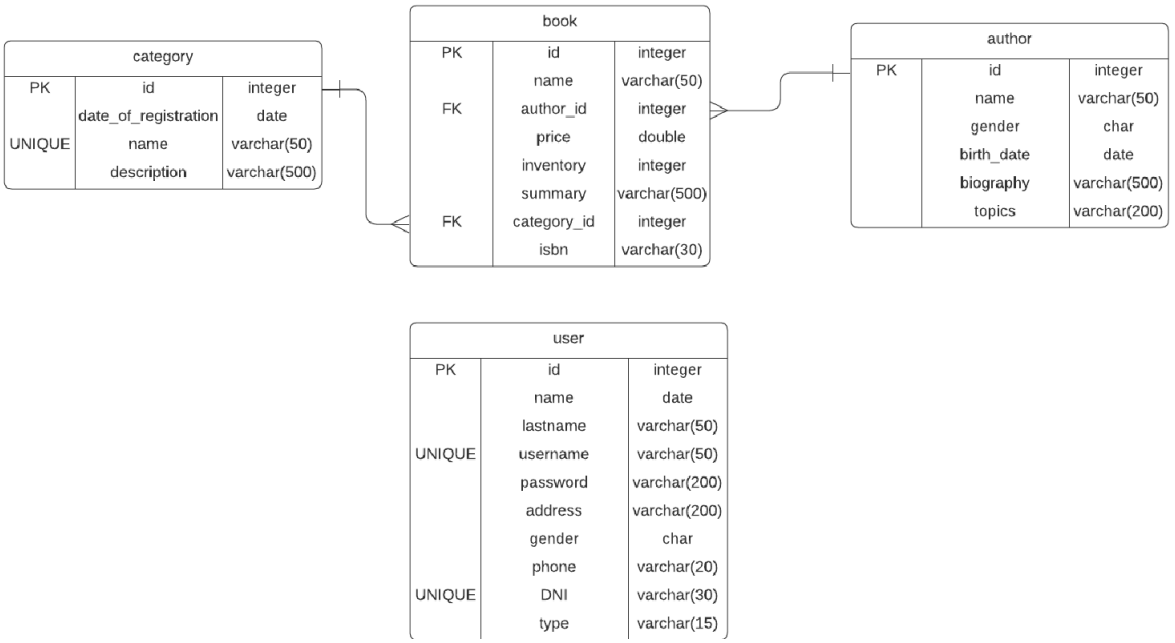
Title	**Author registry**
Primary Actor	Administrator

Success Scenario	After successful login, system will offer a tab for registry. That selection will only be available for admin roles, which will be checked by the authentication service when login occurs. When admin clicks on registry, an author registry tab will be available and when they click on it, a form will be displayed and admin will fill this out and when all details are filled up, registration button will be available to click on it, otherwise admin will be shown at all time a cancel button to abort the registry. When a registry action is confirmed it will send an event to the service in order to send it to the database.

Model and Entities Explanation

In this case the model is composed by the entities book, category, author and user. Although there are many ways to represent the model of the application (a depiction of the state which is persistent in the background of the application and describes completely the real data in the current moment), in this case we are going to take an ER diagram to describe this state.In this case the entity book refers to the abstraction of a book (which is the product offered in the catalog of the library) and has attributes like the id, name (also called title), author_id (which is a reference to the author of the book), price, inventory (which is the number of units of the book which are available still),summary (little text which describes the main idea of the book), category_id (reference to the category which the book belongs), isbn (International Standard Book Number); On the other side, the entity author represents the data of the person who wrote the book and has as attributes the id, name, birth_date,gender and description (which is a little text telling the history of the author and his/her interests), and topics (text with keywords of the author's interest); the category entity which depicts the classification of the books, and has as attributes id, date_of_registration,name and description (explanation of the category); and the last entity is user, which identifies all the users in the system and allow the authentication of them, it has as attributes id, name, lastname, username (which must be unique), password (which will save the hash of the password and not the password in clear text, this practice allow accomplish the functional requirement which says that the password must be saved encrypted), address (the address of the user),gender,phone, DNI (unique identificator of the user) and finally type (which must be customer or administrator).

Basically the catalog of books can be develop based in these entities because we only have to implement the logic of its relationships, a book has an author and is classified in one category, this book can only be registered by an administrator, but any user (customer or administrator can query about it).Also, a category can has many books and also an author can has many books. A user must be authenticated by password and username and has permissions depending on its type: administrator or consumer. As you can see these simple relationships allow both (administrator and customer) to successfully carry out the previously mentioned functional and nonfunctional requirements.

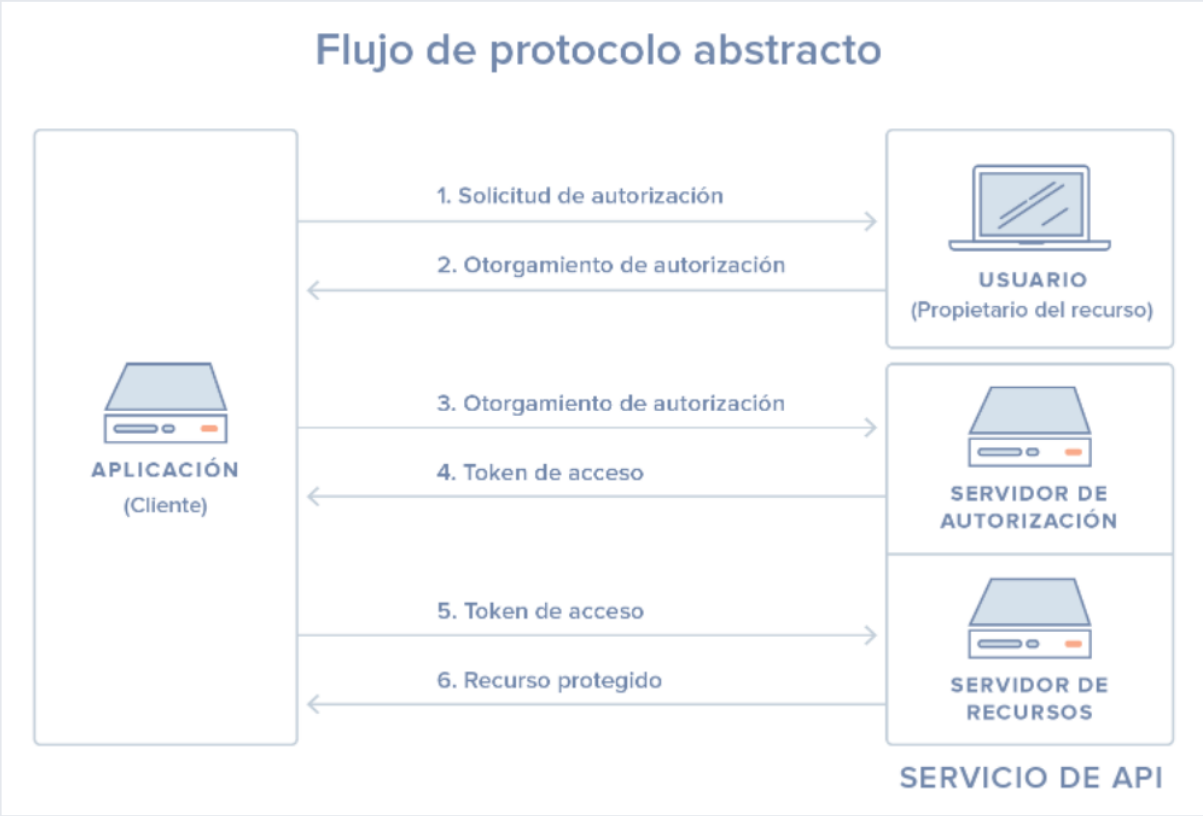


Authentication Flow

In this case we have one endpoint to manage the authentication (the endpoint for user login) which gives a JWT token which will be sent in the authorization header in the following requests to identify the user and validates its permissions. But it is easy to wonder how this process is managed behind the scenes. We are going to describe a scenario which describes how the Oauth2 flow is applied for manage authorization (validation of permissions over actions performed) and authentication (identification of a user in the system), which can be applied behind scenes and let us make clear how the authentication flow would be implemented. Also finally we are going to describe how make this flow more efficient with the help of asymmetric cryptography.

For the administration of user sessions and the authorization of requests, we initially proposed an implementation of the Oauth 2 protocol that is described based on certain roles:

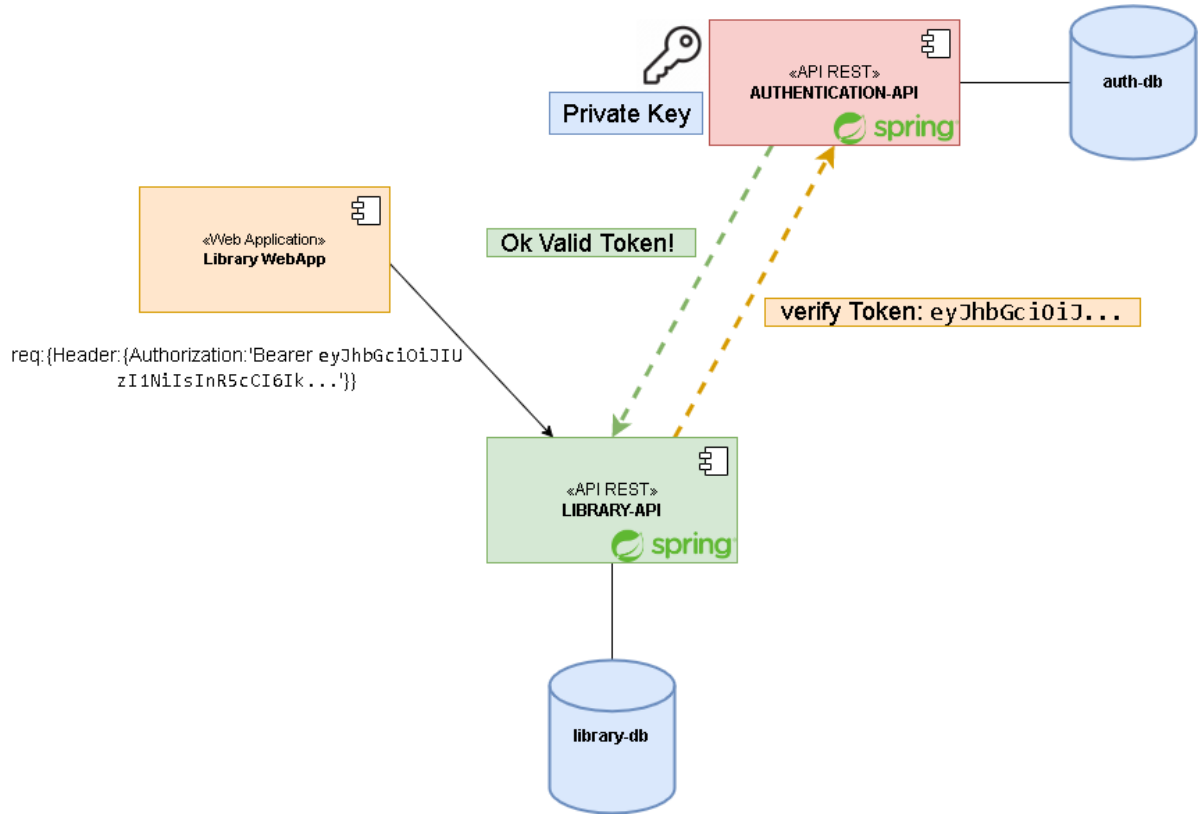
- **Resource Owner:** The resource owner is the user who authorizes an application to access their account.
- **Resource Server**: is the server that hosts the protected resources. (In this case our resources are books, authors, etc)
- **Client**: is the application that you want to access with the credentials of the owner of the resource to perform actions on the system. In this case, the client application would play this role.
- **Authorization Server**: it is the server in charge of authorizing a client given the credentials of the owner of the resource by means of a token, which allows to identify the user who makes the requests in the system and their roles or permissions..



Flow Oauth 2 Protocol

For the implementation of this protocol, an authorization server must be present, described in the architecture as AUTHENTICATION-API. This authorization server receives the credentials of the users when they log in and if they match the registered ones, it returns a JWT-type token. This token is sent in the Authorization header on each request made by the user as a bearer token. In an initial approach, the resource servers would receive the token and make a request to the authorization server to validate that the token was legitimate, since only the authorization server would have the private key with which the tokens were

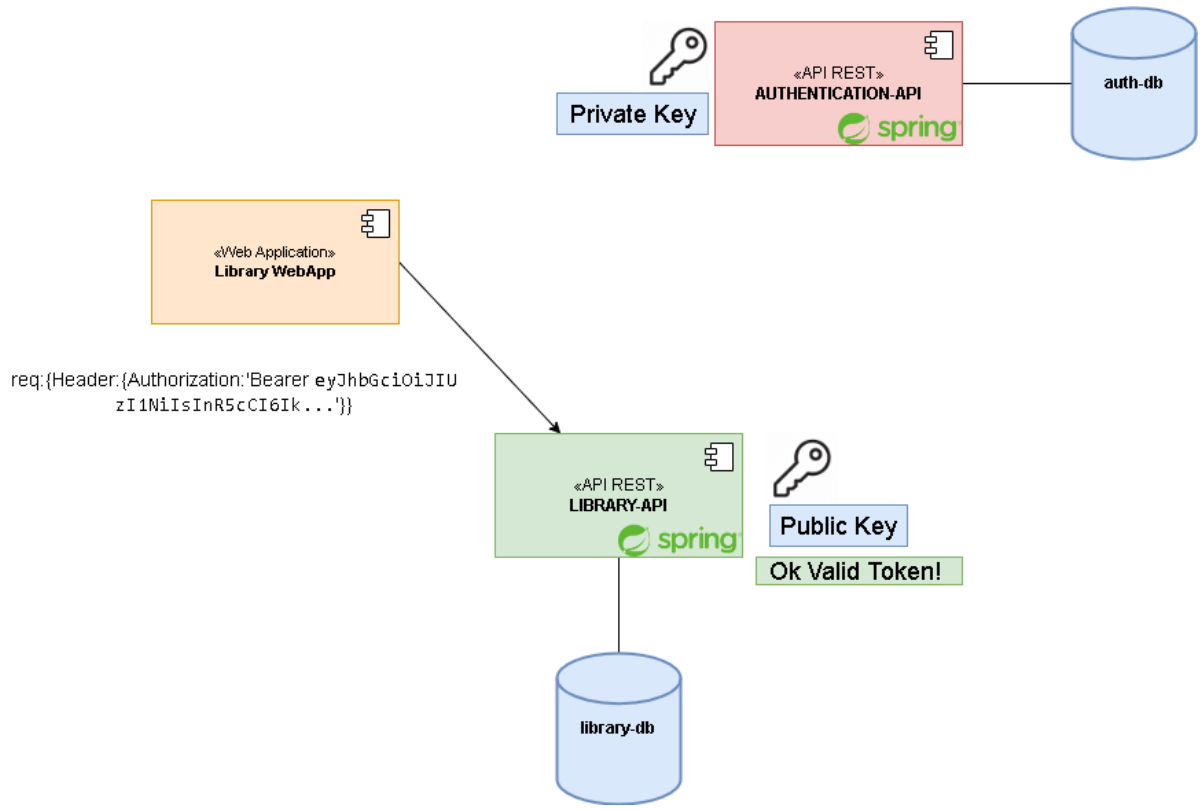
signed. This scenario is illustrated in the next image:



Initial scenario in which the authorization tokens are validated by the authorization server that has a private key to verify its legitimacy.

As can be seen in this scenario, the API is highly dependent on the authorization server, since each time the user makes a request, the API must send another request for the authorization server to verify that the token is legitimate. This dependency increases the level of system coupling and makes the authorization server (in this case the AUTHENTICATION-API component) a single point of failure.

For this reason, a new scenario is proposed in which it is decided to apply public key cryptography to carry out the digital signature of the tokens, for which a pair of keys is generated in the authorization server and when the user logs in the authorization signs the token with the private key. Likewise, when the resource servers (the Library API) start their execution, they consult the public key to the authorization server and store it, in this way, each time it receives a token, they can verify the signature with the public key provided by the server. authorization server without making any additional requests. This allows the Library API to verify token authenticity (integrity) and confidentiality since only the authorization server has the ability to perform token signing. The final scenario (which has less coupling and higher performance) is illustrated in the following image:



Final scenario optimized by the use of public key cryptography: the authorization tokens are validated in the resource servers (Library API) that have a public key to verify their legitimacy.

Operation Description (Endpoints, caching,pagination,XML Representation, JSON Representation and Status Codes)

In this section, all the requests to accomplish the previous requirements are provided. Notice that some responses have the links_related field, which is not part of the model but is used to suggest to the client app how to query for related sources and accomplish the last level of the Richardson Maturity Model. On the other hand, some methods send a cookie to the server to relate the current request with a session_id and in this way make a customized offer of books in the next request to the user. Notice that also are some methods in which the response is cached and this is managed by the cache control header.

Register New User

Protocol: HTTPS
Endpoint: https://library.com/users
URI Design: https://library.com/users
Method: Post
Request Body:

This request body can send in the body the key field only if the register is for an administrator user, this is because the api rest must validate that this key is correct and only allow register users as administrator to authorized staff.

- **JSON Representation:**

```
{
    "name":"Andrés",
    "lastname":"Sanchez",
    "username":"asanchez",
    "password": "password",
    "address":"CRA 12 #65 - 70 Bogotá",
    "gender":"M",
    "phone":"3202000000",
    "DNI":"124548685742",
    "type":"admin",
    "key":"keyRegisT3r4dmiN"
}
```

- **XML Representation:**

```
<?xml version="1.0" encoding="UTF-8" ?>
<root>
  <name>Andrés</name>
  <lastname>Sanchez</lastname>
  <username>asanchez</username>
  <password>password</password>
  <address>CRA 12 #65 - 70 Bogotá</address>
  <gender>M</gender>
  <phone>3202000000</phone>
  <DNI>124548685742</DNI>
  <type>admin</type>
  <key>keyRegisT3r4dmiN</key>
</root>
```

Request Headers:

- **Content-Type:** “application/json” or “application/xml”
- **Host:** library.com

- **Accept:** application/json,application/xml; q=0.9,q=0.8 (preference for application/json response over the application/xml response)
- **Cache-Control:** max-age=0 (don't cache request because is not necessary)
- **Accept-Language**: en-US,en;q=0.8,q=0.5 (preference for american english)
- **User-Agent:** Mozilla/5.0 (Windows NT 10.0; Win64;x64) - or any other data about the requester device.
- **Accept-Encoding:** gzip, deflate, br (content encoding which client can understand)
- **Referer:** https://library.com

Response:

- **HTTP STATUS CODE 201 CREATED:**

  Response Body:

  The response object describes the new state of the object created which also contains the id of the object created in the background database.Notice that in this case the password is not in the response body for security reasons, and even this password is not saved in plain text in the database, instead we must save a hash of the password because is a better security practice on keeping credentials.

  JSON Representation:

```
{
   "id":1,
   "name":"Andrés",
   "lastname":"Sanchez",
   "username":"asanchez",
   "address":"CRA 12 #65 - 70 Bogotá",
   "gender":"M",
   "phone":"3202000000",
   "DNI":"124548685742",
   "type":"admin"
}
```

  XML Representation

```
<?xml version="1.0" encoding="UTF-8" ?>
<root>
  <id>1</id>
  <name>Andrés</name>
  <lastname>Sanchez</lastname>
  <username>asanchez</username>
```

```
  <address>CRA 12 #65 - 70 Bogotá</address>
  <gender>M</gender>
  <phone>3202000000</phone>
  <DNI>124548685742</DNI>
  <type>admin</type>
</root>
```

Response Headers:

server: nginx

date: Wed, 22 Feb 2023 19:31:16 GMT

Content-Type: application/json

Connection: keep-alive (keep the connection alive for next request)

Cache-Control: no-cache (is not necessary cache this response this data is not usually queried)

- **HTTP STATUS CODE 401 UNAUTHORIZED**

  This code is returned when the key value does not match with the key specified in the api for register administrator users.

  JSON Representation:

```
{
"errorCode": "ER6",
"error": "You are not authorized. Insufficient Permissions"
}
```

  XML Representation:

```
<?xml version="1.0" encoding="UTF-8" ?>
<root>
  <errorCode>ER6</errorCode>
  <error>You are not authorized. Insufficient Permissions</error>
</root>
```

- **HTTP STATUS CODE 400 BAD REQUEST**

  This code is returned when the client sends a request body without any of the parameters specified in the representation or with a different type. The specific error and error code is described in the response body, for instance:

  JSON Representation:

```
{
"errorCode": "ER15",
```

"error": "Invalid parameter. name cannot be null"
}

XML Representation:

<?xml version="1.0" encoding="UTF-8" ?>
<root>
<errorCode>ER15</errorCode>
<error>Invalid parameter. name cannot be null</error>
</root>

- **HTTP STATUS CODE 409 CONFLICT:**
  This code is returned when already exists an author with the specified DNI or username.

User Login

Protocol: HTTPS
Endpoint: https://library.com/authentication
URI Design: https://library.com/authentication
Method: Post
Request Body:

- **JSON Representation:**

{
"username":"asanchez",
"password":"password"
}

- **XML Representation:**

<?xml version="1.0" encoding="UTF-8" ?>
<root>
<username>asanchez</username>
<password>password</password>
</root>

Request Headers:

- **Content-Type:** “application/json” or “application/xml”
- **Host:** library.com
- **Accept:** application/json,application/xml; q=0.9,q=0.8 (preference for application/json response over the application/xml response)
- **Cache-Control:** max-age=0 (don't cache request)
- **Accept-Language**: en-US,en;q=0.8,q=0.5 (preference for american english)

- **User-Agent:** Mozilla/5.0 (Windows NT 10.0; Win64;x64) - or any other data about the requester device.
- **Accept-Encoding:** gzip, deflate, br (content encoding which client can understand)
- **Referer:** https://library.com

Response:

- **HTTP STATUS CODE 200:**

  Response Body:

  JSON Representation:


```
{

"token":"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjE2NDg4ODk3MzUsInVzZXJfbmFtZSI6ImFuZHJlc0BoYXNoLmNvIiwiYXV0aG9yaXRpZXMiOlsiUk9MRV9GQVJNRVIiXSwianRpIjoiMmVhYWNiODItMjdiZi00NTExLTg0Y2ItZTc1NmJjZDEyYTNmIiwiY2xpZW50X2lkIjoiY2xpZW50ZSIsInNjb3BlIjpbInJlYWQiLCJ3cml0ZSJdfQ.CHpvjSmGrbp-O1ebAmMVLqMs2H0XO7QgvZNuFppl6zE",
   "role":"administrator"
}
```


  XML Representation


```
<?xml version="1.0" encoding="UTF-8" ?>
<root>

<token>eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjE2NDg4ODk3MzUsInVzZXJfbmFtZSI6ImFuZHJlc0BoYXNoLmNvIiwiYXV0aG9yaXRpZXMiOlsiUk9MRV9GQVJNRVIiXSwianRpIjoiMmVhYWNiODItMjdiZi00NTExLTg0Y2ItZTc1NmJjZDEyYTNmIiwiY2xpZW50X2lkIjoiY2xpZW50ZSIsInNjb3BlIjpbInJlYWQiLCJ3cml0ZSJdfQ.CHpvjSmGrbp-O1ebAmMVLqMs2H0XO7QgvZNuFppl6zE</token>
  <role>administrator</role>
</root>
```


  Response Headers:

  server: nginx

  date: Wed, 22 Feb 2023 19:31:16 GMT

  Content-Type: application/json

  Connection: keep-alive (keep the connection alive for next request)

Cookie: session_id=adv1515a1d; (cookie which identifies a session with the goal of save the last searches of the user in the api and show in the next days results based on this previous searches)

Cache-Control: no-cache (for security is better not to save this response and request a login again after)

- **HTTP STATUS CODE 401 UNAUTHORIZED**

  This code is returned when the credentials are not correct for instance when the password is not correct.The specific error and error code is described in the response body, for instance:

  JSON Representation:

```
{
"errorCode": "ER2",
"error": "Invalid credentials. Incorrect Password"
}
```

  XML Representation:

```
<?xml version="1.0" encoding="UTF-8" ?>
<root>
  <errorCode>ER2</errorCode>
  <error>Invalid credentials. Incorrect Password</error>
</root>
```

- **HTTP STATUS CODE 400 BAD REQUEST**

  This code is returned when the client sends a request body without any of the parameters specified in the representation.

  JSON Representation:

```
{
"errorCode": "ER4",
"error": "Username cannot be null"
}
```

  XML Representation:

```
<?xml version="1.0" encoding="UTF-8" ?>
<root>
  <errorCode>ER4</errorCode>
  <error>Username cannot be null</error>
</root>
```

Register New Book

Protocol: HTTPS
Endpoint: https://library.com/books
URI Desingn: https://library.com/books
Method: Post
Request Body:

- **JSON Representation:**

```
{
   "name":"Life in the Alps",
   "author_id": 1,
   "price": 100000,
   "inventory": 103,
   "summary": "Ben Stiller autobiography where he describes how it is living in the Alps",
   "category_id": 1,
   "isbn": "978-92-95055-02-5"
}
```

- **XML Representation:**

```
<?xml version="1.0" encoding="UTF-8" ?>
<root>
  <name>Life in the Alps</name>
  <author_id>1</author_id>
  <price>100000</price>
  <inventory>103</inventory>
  <summary>Ben Stiller autobiography where he describes how it is living in the Alps</summary>
  <category_id>1</category_id>
  <isbn>978-92-95055-02-5</isbn>
</root>
```

Request Headers:

- **Content-Type:** “application/json” or “application/xml”
- **Host:** library.com
- **Accept:** application/json,application/xml; q=0.9,q=0.8 (preference for application/json response over the application/xml response)
- **Cache-Control:** max-age=0 (don't cache request)

- **Accept-Language**: en-US,en;q=0.8,q=0.5 (preference for american english)
- **User-Agent:** Mozilla/5.0 (Windows NT 10.0; Win64;x64) - or any other data about the requester device.
- **Accept-Encoding:** gzip, deflate, br (content encoding which client can understand)
- **Referer:** https://library.com
- **Authorization: "**Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjE2NDg4ODk3MzUsInVzZXJfbmFtZSI6ImFuZHJlc0BoYXNoLmNvIiwiYXV0aG9yaXRpZXMiOlsiUk9MRV9GQVJNRVIiXSwianRpIjoiMmVhYWNiODItMjdiZi00NTExLTg0Y2ItZTc1NmJjZDEyYTNmIiwiY2xpZW50X2lkIjoiY2xpZW50ZSIsInNjb3BlIjpbInJlYWQiLCJ3cml0ZSJdfQ.CHpvjSmGrbp-O1ebAmMVLqMs2H0XO7QgvZNuFppl6zE" (a bearer token to identify the user who is requesting and verify if has the permission needed)

Response:

- **HTTP STATUS CODE 201 CREATED:**

  Response Body:

  The response object describes the new state of the object created which also contains the id of the object created in the background database.

  JSON Representation:


```
{
   "id":1,
   "name":"Life in the Alps",
   "author_id": 1,
   "price": 100000,
   "inventory": 103,
   "summary": "Ben Stiller autobiography where he describes how it is living in the Alps",
   "category_id": 1,
   "isbn": "978-92-95055-02-5" ,
"links_related":["https://library.com/books?name=Life in the Alps","https://library.com/books?category_id=1"]
}
```

XML Representation

```
<?xml version="1.0" encoding="UTF-8" ?>
<root>
 <id>1</id>
 <name>Life in the Alps</name>
 <author_id>1</author_id>
 <price>100000</price>
 <inventory>103</inventory>
 <summary>Ben Stiller autobiography where he describes how it is living in the Alps</summary>
 <category_id>1</category_id>
 <isbn>978-92-95055-02-5</isbn>
 <links_related>https://library.com/books?name=Life in the Alps</links_related>
 <links_related>https://library.com/books?category_id=1</links_related>
</root>
```

Response Headers:

server: nginx

date: Wed, 22 Feb 2023 19:31:16 GMT

Content-Type: application/json

Connection: keep-alive (keep the connection alive for next request)

Cache-Control: no-cache (is not necessary cache this response because the object is created only one time and after is only queried)

- **HTTP STATUS CODE 401 UNAUTHORIZED**

  This code is returned when the token provided in the authorization header has expired, is not valid or the user does not have the necessary permissions (to be an administrator) to carry out the request. The specific error and error code is described in the response body, for instance:

  JSON Representation:

```
{
"errorCode": "ER1",
```

```
"error": "You are not authorized. Insufficient Permissions"
}
```

XML Representation:

```
<?xml version="1.0" encoding="UTF-8" ?>
<root>
  <errorCode>ER1</errorCode>
  <error>You are not authorized. Insufficient Permissions</error>
</root>
```

- **HTTP STATUS CODE 400 BAD REQUEST**
  This code is returned when the client sends a request body without any of the parameters specified in the representation or with a different type. The specific error and error code is described in the response body, for instance:

  JSON Representation:

```
{
"errorCode": "ER12",
"error": "Invalid parameter. price must be a number"
}
```

  XML Representation:

```
<?xml version="1.0" encoding="UTF-8" ?>
<root>
  <errorCode>ER12</errorCode>
  <error>Invalid parameter. price must be a number</error>
</root>
```

- **HTTP STATUS CODE 409 CONFLICT:**
  This code is returned when the book in the body has the same isbn of a registered book.

Delete a Book

Protocol: HTTPS
Endpoint: https://library.com/books
URI Desingn: https://library.com/books/{id_book}
Method: Delete
Request Path Variable:

- **id_book (mandatory):** id of the book to be deleted.

Request Headers:

- **Content-Type:** “application/json” or “application/xml”

- **Host:** library.com
- **Accept:** application/json,application/xml; q=0.9,q=0.8 (preference for application/json response over the application/xml response)
- **Cache-Control:** max-age=0 (don't cache request because the data returned by this delete will not be queried after)
- **Accept-Language**: en-US,en;q=0.8,q=0.5 (preference for american english)
- **User-Agent:** Mozilla/5.0 (Windows NT 10.0; Win64;x64) - or any other data about the requester device.
- **Accept-Encoding:** gzip, deflate, br (content encoding which client can understand)
- **Referer:** https://library.com
- **Authorization: "**Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjE2NDg4ODk3MzUsInVzZXJfbmFtZSI6ImFuZHJlc0BoYXNoLmNvIiwiYXV0aG9yaXRpZXMiOlsiUk9MRV9GQVJNRVIiXSwianRpIjoiMmVhYWNiODItMjdiZi00NTExLTg0Y2ItZTc1NmJjZDEyYTNmIiwiY2xpZW50X2lkIjoiY2xpZW50ZSIsInNjb3BlIjpbInJlYWQiLCJ3cml0ZSJdfQ.CHpvjSmGrbp-O1ebAmMVLqMs2H0XO7QgvZNuFppl6zE" (a bearer token must belong to an administrator user in order to delete the book)

Response:

- **HTTP STATUS CODE 200 OK:**

  Response Body:

  The response object in this case doesn't have a body because the new state of the application doesn't have the deleted resource. The status 200 means that the resource (in this case the book) has been deleted successfully.

  Response Headers:

  server: nginx

  date: Wed, 22 Feb 2023 19:31:16 GMT

  Content-Type: application/json

  Connection: keep-alive (keep the connection alive for next request)

  Cache-Control: no-cache (is not necessary cache this response because the object was deleted and there is not any interest in make this request again)

- **HTTP STATUS CODE 401 UNAUTHORIZED**

  This code is returned when the token provided in the authorization header has expired, is not valid or the user does not have the necessary permissions (to be an administrator in this case) to carry out the request. The specific error and error code is described in the response body, for instance:

  JSON Representation:

```
{
"errorCode": "ER1",
"error": "You are not authorized. Insufficient Permissions"
}
```

  XML Representation:

```
<?xml version="1.0" encoding="UTF-8" ?>
<root>
  <errorCode>ER1</errorCode>
  <error>You are not authorized. Insufficient Permissions</error>
</root>
```

- **HTTP STATUS CODE 400 BAD REQUEST**

  This code is returned when the client sends the request without specifying the book_id. The specific error and error code is described in the response body, for instance:

JSON Representation:


```
{
"errorCode": "ER14",
"error": "Invalid parameter. id_book cannot be null"
}
```


XML Representation:


```
<?xml version="1.0" encoding="UTF-8" ?>
<root>
  <errorCode>ER14</errorCode>
  <error>Invalid parameter. id_book cannot be null</error>
</root>
```


- **HTTP STATUS CODE 409 CONFLICT:**
  This code is returned when the book with the id_book provided doesn't exist, then is not found and cannot be deleted in the database, causing a conflict.

Update a Book

Protocol: HTTPS
Endpoint: https://library.com/books
URI Desingn: https://library.com/books
Method: Put (We can also consider this as a patch if we want to limit the fields that can be changed but to be brief we prefer replace the entire object and implement this method like a put request, for this reason the administrator has the power to decide even fields which don't should be changed like isbn and name of the book)
Request Body:

- **JSON Representation:**


```
{
        "id":1,
    "name":"Life in the Mountains",
    "author_id": 1,
    "price": 750000,
    "inventory": 25,
    "summary": "Ben Stiller autobiography where he describes how it is living in the mountains",
    "category_id": 1,
```

```
    "isbn": "978-92-95055-02-5"

}
```


- **XML Representation:**


```
<?xml version="1.0" encoding="UTF-8" ?>
<root>
  <id>1</id>
  <name>Life in the Mountains</name>
  <author_id>1</author_id>
  <price>750000</price>
  <inventory>25</inventory>
  <summary>Ben Stiller autobiography where he describes how it is living in the mountains</summary>
  <category_id>1</category_id>
  <isbn>978-92-95055-02-5</isbn>
 </root>
```


Request Headers:

- **Content-Type:** “application/json” or “application/xml”
- **Host:** library.com
- **Accept:** application/json,application/xml; q=0.9,q=0.8 (preference for application/json response over the application/xml response)
- **Cache-Control:** max-age=0 (don't cache request because update is not a recurrent request)
- **Accept-Language**: en-US,en;q=0.8,q=0.5 (preference for american english)
- **User-Agent:** Mozilla/5.0 (Windows NT 10.0; Win64;x64) - or any other data about the requester device.
- **Accept-Encoding:** gzip, deflate, br (content encoding which client can understand)
- **Referer:** https://library.com

- **Authorization: “**Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjE2NDg4ODk3MzUsInVzZXJfbmFtZSI6ImFuZHJlc0BoYXNoLmNvIiwiYXV0aG9yaXRpZXMiOlsiUk9MRV9GQVJNRVIiXSwianRpIjoiMmVhYWNiODItMjdiZi00NTExLTg0Y2ItZTc1NmJjZDEyYTNmIiwiY2xpZW50X2lkIjoiY2xpZW50ZSIsInNjb3BlIjpbInJlYWQiLCJ3cml0ZSJdfQ.CHpvjSmGrbp-O1ebAmMVLqMs2H0XO7QgvZNuFppl6zE” (a bearer token which must belongs to an administrator user)

Response:

- **HTTP STATUS CODE 200 OK:**

  Response Body:

  The response object describes the new state of the object which replaces the previous state (the previous book registered with the id specified in the request).

  JSON Representation:


```
{
        "id":1,
   "name":"Life in the Mountains",
   "author_id": 1,
   "price": 750000,
   "inventory": 25,
   "summary": "Ben Stiller autobiography where he describes how it is living in the mountains",
   "category_id": 1,
   "isbn": "978-92-95055-02-5",
"links_related":["https://library.com/books?name=Life in the Mountains","https://library.com/books?category_id=1"]
}
```


  XML Representation


```
<?xml version="1.0" encoding="UTF-8" ?>
<root>
  <id>1</id>
  <name>Life in the Mountains</name>
  <author_id>1</author_id>
  <price>750000</price>
  <inventory>25</inventory>
  <summary>Ben Stiller autobiography where he describes how it is living in the mountains</summary>
  <category_id>1</category_id>
  <isbn>978-92-95055-02-5</isbn>
<links_related>https://library.com/books?name=Life in the Mountains</links_related>
 <links_related>https://library.com/books?category_id=1</links_related>
```

```
</root>
```

Response Headers:

server: nginx

date: Wed, 22 Feb 2023 19:31:16 GMT

Content-Type: application/json

Connection: keep-alive (keep the connection alive for next request)

Cache-Control: no-cache (is not necessary cache this response because the update of a book is not a recurrent request)

- **HTTP STATUS CODE 401 UNAUTHORIZED**

  This code is returned when the token provided in the authorization header has expired, is not valid or the user does not have the necessary permissions (to be an administrator) to carry out the request. The specific error and error code is described in the response body, for instance:

  JSON Representation:

```
{
"errorCode": "ER1",
"error": "You are not authorized. Insufficient Permissions"
}
```

  XML Representation:

```
<?xml version="1.0" encoding="UTF-8" ?>
<root>
  <errorCode>ER1</errorCode>
  <error>You are not authorized. Insufficient Permissions</error>
</root>
```

- **HTTP STATUS CODE 400 BAD REQUEST**

  This code is returned when the client sends a request body without any of the parameters specified in the representation or with a different type and the application

cannot allow change a state for a new inconsistent state. The specific error and error code is described in the response body, for instance:

JSON Representation:

```
{
"errorCode": "ER12",
"error": "Invalid request. id cannot be null"
}
```

XML Representation:

```
<?xml version="1.0" encoding="UTF-8" ?>
<root>
  <errorCode>ER12</errorCode>
  <error>Invalid request. id cannot be null</error>
</root>
```

- **HTTP STATUS CODE 409 CONFLICT:**
  This code is returned when the book with the specified id does not exist, the author with the author_id specified doesn't exist or the category with the category_id does not exist or already exists a book with the specified isbn.The specific error and error code is described in the response body, for instance:

  JSON Representation:

  ```
  {
  "errorCode": "ER21",
  "error": "Conflict. the author specified doesn't exist"
  }
  ```

  XML Representation:

  ```
  <?xml version="1.0" encoding="UTF-8" ?>
  <root>
    <errorCode>ER21</errorCode>
    <error>Conflict. the author specified doesnt exist</error>
  </root>
  ```

Register New Author

Protocol: HTTPS
Endpoint: https://library.com/authors
URI Desingn: https://library.com/authors
Method: Post

Request Body:

- **JSON Representation:**

```
{
"name":"Ben Stiller",
"country":"USA",
"gender":"M",
"birth_date":"30/11/1965",
"biography": "He was Hollywood star who now dedicates himself to tell stories and inspiring events that happened during his time in movies",
"topics": "Adventure and Drama"
}
```

- **XML Representation:**

```
<?xml version="1.0" encoding="UTF-8" ?>
<root>
  <name>Ben Stiller</name>
  <country>USA</country>
  <gender>M</gender>
  <birth_date>30/11/1965</birth_date>
  <biography>He was Hollywood star who now dedicates himself to tell stories and inspiring events that happened during his time in movies</biography>
  <topics>Adventure and Drama</topics>

</root>
```

Request Headers:

- **Content-Type:** “application/json” or “application/xml”
- **Host:** library.com
- **Accept:** application/json,application/xml; q=0.9,q=0.8 (preference for application/json response over the application/xml response)
- **Cache-Control:** max-age=0 (don't cache request because creation of author is not a recurrent request)
- **Accept-Language**: en-US,en;q=0.8,q=0.5 (preference for american english)

- **User-Agent:** Mozilla/5.0 (Windows NT 10.0; Win64;x64) - or any other data about the requester device.
- **Accept-Encoding:** gzip, deflate, br (content encoding which client can understand)
- **Referer:** https://library.com
- **Authorization: "**Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjE2NDg4ODk3MzUsInVzZXJfbmFtZSI6ImFuZHJlc0BoYXNoLmNvIiwiYXV0aG9yaXRpZXMiOlsiUk9MRV9GQVJNRVIiXSwianRpIjoiMmVhYWNiODItMjdiZi00NTExLTg0Y2ItZTc1NmJjZDEyYTNmIiwiY2xpZW50X2lkIjoiY2xpZW50ZSIsInNjb3BlIjpbInJlYWQiLCJ3cml0ZSJdfQ.CHpvjSmGrbp-O1ebAmMVLqMs2H0XO7QgvZNuFppl6zE" (a bearer token to identify the user who is requesting and verify if has the permission needed, in this case the user must be an administrator to register a new author)

Response:

- **HTTP STATUS CODE 201 CREATED:**

  Response Body:

  The response object describes the new state of the object created which also contains the id of the object created in the background database.

  JSON Representation:

```
{
   "id":1,
"name":"Ben Stiller",
"country":"USA",
"gender":"M",
"birth_date":"30/11/1965",
"biography": "He was Hollywood star who now dedicates himself to tell stories and inspiring events that happened during his time in movies",
"topics": "Adventure and Drama",
"links_related": ["https://library.com/authors?name=Ben Stiller"]
}
```

  XML Representation

```
<?xml version="1.0" encoding="UTF-8" ?>
<root>
```

<id>1</id>
<name>Ben Stiller</name>
<country>USA</country>
<gender>M</gender>
<birth_date>30/11/1965</birth_date>
<biography>He was Hollywood star who now dedicates himself to tell stories and inspiring events that happened during his time in movies</biography>
<topics>Adventure and Drama</topics>
<links_related>https://library.com/authors?name=Ben Stiller</links_related>
</root>

Response Headers:

server: nginx

date: Wed, 22 Feb 2023 19:31:16 GMT

Content-Type: application/json

Connection: keep-alive (keep the connection alive for next request)

Cache-Control: no-cache (is not necessary cache this response because the object is created only one time and after is only queried)

- **HTTP STATUS CODE 401 UNAUTHORIZED**

  This code is returned when the token provided in the authorization header has expired, is not valid or the user does not have the necessary permissions (to be an administrator) to carry out the request. The specific error and error code is described in the response body, for instance:

  JSON Representation:

  ```
  {
  "errorCode": "ER1",
  "error": "You are not authorized. Insufficient Permissions"
  }
  ```

  XML Representation:

  ```
  <?xml version="1.0" encoding="UTF-8" ?>
  <root>
  <errorCode>ER1</errorCode>
  <error>You are not authorized. Insufficient Permissions</error>
  </root>
  ```

- **HTTP STATUS CODE 400 BAD REQUEST**
  This code is returned when the client sends a request body without any of the parameters specified in the representation or with a different type. The specific error and error code is described in the response body, for instance:

  JSON Representation:

  ```
  {
  "errorCode": "ER12",
  "error": "Invalid parameter. Country cannot be null"
  }
  ```

  XML Representation:

  ```
  <?xml version="1.0" encoding="UTF-8" ?>
  <root>
    <errorCode>ER12</errorCode>
    <error>Invalid parameter.Country cannot be null </error>
  </root>
  ```

Register New Category

Protocol: HTTPS
Endpoint: https://library.com/categories
URI Design: https://library.com/categories
Method: Post
Request Body:
This body will have two fields one for the category name to be registered and the description for it, when sending this request name attribute will be checked in order to validate duplicate categories. It is to be noticed that an authorization header will be sent in order to confirm the user has an admin role to make registers on this database.

- **JSON Representation:**

  ```
  {
     "name":"Action",
     "description":"This topic is meant for stories where battles happen and where the main character confronts dangerous situation"
  }
  ```

- **XML Representation:**

  ```
  <?xml version="1.0" encoding="UTF-8" ?>
  <root>
    <name>Action</name>
    <description>
  ```

```
        This topic is meant for stories where battles happen and where the main character confronts dangerous situation
    </description>
</root>
```

Request Headers:

- **Content-Type:** “application/json” or “application/xml”
- **Host:** library.com
- **Accept:** application/json,application/xml; q=0.9,q=0.8 (preference for application/json response over the application/xml response)
- **Cache-Control:** max-age=0 (don't cache request because is not necessary)
- **Accept-Language**: en-US,en;q=0.8,q=0.5 (preference for american english)
- **User-Agent:** Mozilla/5.0 (Windows NT 10.0; Win64;x64) - or any other data about the requester device.
- **Accept-Encoding:** gzip, deflate, br (content encoding which client can understand)
- **Referer:** https://library.com
- **Authorization: “**Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjE2NDg4ODk3MzUsInVzZXJfbmFtZSI6ImFuZHJlc0BoYXNoLmNvIiwiYXV0aG9yaXRpZXMiOlsiUk9MRV9GQVJNRVIiXSwianRpIjoiMmVhYWNiODItMjdiZi00NTExLTg0Y2ItZTc1NmJjZDEyYTNmIiwiY2xpZW50X2lkIjoiY2xpZW50ZSIsInNjb3BlIjpbInJlYWQiLCJ3cml0ZSJdfQ.CHpvjSmGrbp-O1ebAmMVLqMs2H0XO7QgvZNuFppl6zE” (this token can belong only to an administrator)

Response:

- **HTTP STATUS CODE 201 CREATED:**

  Response Body:

  The response object describes the new state of the object created which also contains the id of the object created in the background database.

  JSON Representation:

```
{
   "id":3,
   "date_of_registration": "2023-02-26T22:56:02.038Z",
   "name":"Action",
   "description":"This topic is meant for stories where battles happen and where the main character confronts dangerous situation",
"links_related":["https://library.com/categories?category_id=3"]
}
```

XML Representation

```
<?xml version="1.0" encoding="UTF-8" ?>
<root>
  <id>3</id>
  <date_of_registration>2023-02-26T22:56:02.038Z</date_of_registration>
  <name>Action</name>
  <description>This topic is meant for stories where battles happen and where the main character confronts dangerous situation</description>
  <links_related>https://library.com/categories?category_id=3</links_related>
</root>
```

Response Headers:

server: nginx

date: Wed, 22 Feb 2023 19:31:16 GMT

Content-Type: application/json

Connection: keep-alive (keep the connection alive for next request)

Cache-Control: no-cache (is not necessary cache this response this data is not usually queried)

- **HTTP STATUS CODE 401 UNAUTHORIZED**

  This code is returned when the header value does not match with any user for admin role or token has expired.

  JSON Representation:

```
{
    "errorCode": "ER6",
    "error": "You are not authorized. Insufficient Permissions”
}
```

  XML Representation:

```
<?xml version="1.0" encoding="UTF-8" ?>
<root>
    <errorCode>ER6</errorCode>
    <error>You are not authorized. Insufficient Permissions
    </error>
</root>
```

- **HTTP STATUS CODE 400 BAD REQUEST**

This code is returned when the client sends a request body without any of the parameters specified in the representation or with a different type. The specific error and error code is described in the response body, for instance:

JSON Representation:

```
{
        "errorCode": "ER15",
        "error": "Invalid parameter. name cannot be null"
}
```

XML Representation:

```
<?xml version="1.0" encoding="UTF-8" ?>
<root>
  <errorCode>ER15</errorCode>
  <error>Invalid parameter. name cannot be null</error>
</root>
```

- **HTTP STATUS CODE 409 CONFLICT:**
  This code is returned when the new category has a name of category which is already registered.

Get Books

Protocol: HTTPS
Endpoint: https://library.com/books
URI Desingn:
https://library.com/books?name=<name>&author_id=<author_id>&category_id<category_id>&offset=<offset>&limit=<limit>
Method: Get
Query parameters:

- **name (optional):** string which must be contained in the name of the book
- **author_id (optional):** id of the author of the books queried.
- **category_id (optional):** id of the category which belongs the books.
- **offset (optional):** offset of the query (which let the application apply pagination).
- **limit (optional):** number of books expected of the query.

Notice that all the query parameters are optional, in case no one of them is filled all the books will be retrieved and the offset and limit parameters will be defined with values by default.

Request Headers:

- **Content-Type:** “application/json” or “application/xml”
- **Host:** library.com

- **Accept:** application/json,application/xml; q=0.9,q=0.8 (preference for application/json response over the application/xml response)
- **Cache-Control:** max-age=1800 (This request will be cached because is recurrent, it means that can be executed multiple times in a short period of time when the user is searching for books, and for instance, the user can query the second page of a search result, next switch to the third page and after return to the second page, then caching is necessary and executed)
- **Accept-Language**: en-US,en;q=0.8,q=0.5 (preference for american english)
- **User-Agent:** Mozilla/5.0 (Windows NT 10.0; Win64;x64) - or any other data about the requester device.
- **Accept-Encoding:** gzip, deflate, br (content encoding which client can understand)
- **Referer:** https://library.com
- **Authorization: "**Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjE2NDg4ODk3MzUsInVzZXJfbmFtZSI6ImFuZHJlc0BoYXNoLmNvIiwiYXV0aG9yaXRpZXMiOlsiUk9MRV9GQVJNRVIiXSwianRpIjoiMmVhYWNiODItMjdiZi00NTExLTg0Y2ItZTc1NmJjZDEyYTNmIiwiY2xpZW50X2lkIjoiY2xpZW50ZSIsInNjb3BlIjpbInJlYWQiLCJ3cml0ZSJdfQ.CHpvjSmGrbp-O1ebAmMVLqMs2H0XO7QgvZNuFppl6zE" (this token can belong to a administrator or a customer)
- **Cookie:** session_id=adv1515a1d; (this header is used to save in the server the last books queried for a user and the categories which are of his/her interest, this makes the system able to show a best offer to the user and for example put in the first page the books queried previously)

Response:

- **HTTP STATUS CODE 200 OK:**

  Response Body:

  The response object in this case represents the state of the books queried by the user. The status 200 means that the resource (in this case a list of books) has been retrieved successfully.

  JSON Representation:


```
[{
   "id":1,
   "name":"Life in the Alps",
   "author_id": 1,
   "price": 100000,
   "inventory": 103,
   "summary": "Ben Stiller autobiography where he describes how it is living in the Alps",
   "category_id": 1,
   "isbn": "978-92-95055-02-5" ,
  "links_related": ["https://library.com/authors?name=Ben Stiller","https://library.com/categories?category_id=1"]
},
{
   "id":2,
   "name":"1984",
   "author_id": 2,
   "price": 380000,
   "inventory": 80,
   "summary": "In the future world of 1984, the world is divided up into three superstates—Oceania, Eurasia, and Eastasia—that are deadlocked in a permanent war. The superpowers are so evenly matched that a decisive victory is impossible, but the real reason for the war is to keep their economies productive without adding to the wealth of their citizens",
   "category_id": 2,
   "isbn": "947-92-95485-03-5" ,
  "links_related": ["https://library.com/authors?name=George Orwell","https://library.com/categories?category_id=2"]
}...]
```

XML Representation:

```
<?xml version="1.0" encoding="UTF-8" ?>
<root>
  <row>
    <id>1</id>
    <name>Life in the Alps</name>
    <author_id>1</author_id>
    <price>100000</price>
    <inventory>103</inventory>
    <summary>Ben Stiller autobiography where he describes how it is living in the Alps</summary>
    <category_id>1</category_id>
    <isbn>978-92-95055-02-5</isbn>
    <links_related>https://library.com/authors?name=Ben Stiller</links_related>

<links_related>https://library.com/categories?category_id=1</links_related>
  </row>
  <row>
    <id>2</id>
    <name>1984</name>
    <author_id>2</author_id>
    <price>380000</price>
    <inventory>80</inventory>
    <summary>In the future world of 1984, the world is divided up into three superstates—Oceania, Eurasia, and Eastasia—that are deadlocked in a permanent war. The superpowers are so evenly matched that a decisive victory is impossible, but the real reason for the war is to keep their economies productive without adding to the wealth of their citizens</summary>
    <category_id>2</category_id>
    <isbn>947-92-95485-03-5</isbn>
    <links_related>https://library.com/authors?name=George Orwell</links_related>

<links_related>https://library.com/categories?category_id=2</links_related>
  </row>
  …
```

```
</root>
```

Response Headers:

server: nginx

date: Wed, 22 Feb 2023 19:31:16 GMT

Content-Type: application/json

Connection: keep-alive (keep the connection alive for next request)

Cache-Control: max-age=1800 (This request will be cached because is recurrent, it means that can be executed multiple times in a short period of time when the user is searching for books, and for instance, the user can query the second page of a search result, next switch to the third page and after return to the second page, then caching is necessary and executed)

- **HTTP STATUS CODE 401 UNAUTHORIZED**

  This code is returned when the token provided in the authorization header has expired or is not valid to carry out the request. The specific error and error code is described in the response body, for instance:

  JSON Representation:

```
{
"errorCode": "ER1",
"error": "You are not authorized. Insufficient Permissions"
}
```

  XML Representation:

```
<?xml version="1.0" encoding="UTF-8" ?>
<root>
  <errorCode>ER1</errorCode>
  <error>You are not authorized. Insufficient Permissions</error>
</root>
```

- **HTTP STATUS CODE 400 BAD REQUEST**

  This code is returned when the client sends the request without the cookie session_id . The specific error and error code is described in the response body, for instance:

JSON Representation:

```
{
"errorCode": "ER14",
"error": "Invalid parameter. session_id cannot be null"
}
```

XML Representation:

```
<?xml version="1.0" encoding="UTF-8" ?>
<root>
  <errorCode>ER14</errorCode>
  <error>Invalid parameter. session_id cannot be null</error>
</root>
```

- **HTTP STATUS CODE 409 CONFLICT:**
  This code is returned when the category with category_id doesn't exist or when the author with author_id doesn't exist.

Get Authors

Protocol: HTTPS

Endpoint: https://library.com/authors
URI Desingn: https://library.com/authors?name=<name>&offset=<offset>&limit=<limit>
Method: Get
Query parameters:

- **name (optional):** string which must be contained in the name of the author
- **offset (optional):** offset of the query (which let the application apply pagination).
- **limit (optional):** number of authors expected of the query.

Notice that all the query parameters are optional, in case none of them is filled all the authors will be retrieved and the offset and limit parameters will be defined with values by default.
Request Headers:

- **Content-Type:** "application/json" or "application/xml"
- **Host:** library.com
- **Accept:** application/json,application/xml; q=0.9,q=0.8 (preference for application/json response over the application/xml response)
- **Cache-Control:** max-age=3600 (This request will be cached because is recurrent, it means that can be executed multiple times in a short period of time when the user is searching for authors, and for instance, the user can query the second page of a

search result, next switch to the third page and after return to the second page, then caching is necessary and executed)

- **Accept-Language**: en-US,en;q=0.8,q=0.5 (preference for american english)
- **User-Agent:** Mozilla/5.0 (Windows NT 10.0; Win64;x64) - or any other data about the requester device.
- **Accept-Encoding:** gzip, deflate, br (content encoding which client can understand)
- **Referer:** https://library.com
- **Authorization: "**Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjE2NDg4ODk3MzUsInVzZXJfbmFtZSI6ImFuZHJlc0BoYXNoLmNvIiwiYXV0aG9yaXRpZXMiOlsiUk9MRV9GQVJNRVIiXSwianRpIjoiMmVhYWNiODItMjdiZi00NTExLTg0Y2ItZTc1NmJjZDEyYTNmIiwiY2xpZW50X2lkIjoiY2xpZW50ZSIsInNjb3BlIjpbInJlYWQiLCJ3cml0ZSJdfQ.CHpvjSmGrbp-O1ebAmMVLqMs2H0XO7QgvZNuFppl6zE" (this token can belong to a administrator or a customer)

Response:

- **HTTP STATUS CODE 200 OK:**

  Response Body:

  The response object in this case represents the state of the authors queried by the user. The status 200 means that the resource (in this case a list of authors) has been retrieved successfully.

  JSON Representation:


```
[
    {
        "id":1,
        "name":"Ben Stiller",
        "country":"USA",
        "gender":"M",
        "birth_date":"30/11/1965",
        "biography": "He was Hollywood star who now dedicates himself to tell stories and inspiring events that happened during his time in movies",
        "topics": ["Adventure", "Drama"]
    },
    {
        "id":2,
```

```
                "name":"George Orwell",
                "country":"USA",
                "gender":"M",
                "birth_date":"30/11/1965",
                "biography": "He is author of famous book 1984",
                "topics": ["Drama", "Humour"]
            }
    ]
```


XML Representation:


```
<?xml version="1.0" encoding="UTF-8" ?>
<root>
 <row>
  <id>1</id>
  <name>Ben Stiller</name>
  <country>USA</country>
  <gender>M</gender>
  <birth_date>30/11/1965</birth_date>
  <biography>He was Hollywood star who now dedicates himself to tell stories and inspiring events that happened during his time in movies</biography>
  <topics>Adventure</topics>
  <topics>Drama</topics>
 </row>
 <row>
  <id>2</id>
  <name>George Orwell</name>
  <country>USA</country>
  <gender>M</gender>
  <birth_date>30/11/1965</birth_date>
  <biography>He is author of famous book 1984</biography>
  <topics>Drama</topics>
  <topics>Humour</topics>
 </row>
</root>
```


Response Headers:

server: nginx

date: Wed, 22 Feb 2023 19:31:16 GMT

Content-Type: application/json

Connection: keep-alive (keep the connection alive for next request)

Cache-Control: max-age=3600 (This request will be cached because is recurrent, it means that can be executed multiple times in a short period of time when the user is registering a book and needs to specify the author of the book)

- **HTTP STATUS CODE 401 UNAUTHORIZED**

  This code is returned when the token provided in the authorization header has expired or is not valid to carry out the request. The specific error and error code is described in the response body, for instance:

  JSON Representation:

```
{
    "errorCode": "ER1",
    "error": "You are not authorized. Insufficient Permissions"
}
```

  XML Representation:

```
<?xml version="1.0" encoding="UTF-8" ?>
<root>
  <errorCode>ER1</errorCode>
  <error>You are not authorized. Insufficient Permissions</error>
</root>
```

- **HTTP STATUS CODE 404 NOT FOUND:**

  This code is returned when the author with name specified doesn't exist.

Get Categories

Protocol: HTTPS
Endpoint: https://library.com/categories
URI Desingn: https://library.com/categories?category_id=<category_id>
Method: Get
Query parameters:

- **category_id (optional):** id of the category.

Notice that if the query parameter "category_id" is not specified then all the categories will be retrieved (without pagination because this query is important to register other entities like books and a paginated query will not accomplish in a suitable way this requirement)
Request Headers:

- **Content-Type:** "application/json" or "application/xml"
- **Host:** library.com
- **Accept:** application/json,application/xml; q=0.9,q=0.8 (preference for application/json response over the application/xml response)

- **Cache-Control:** max-age=1800 (This request will be cached because is recurrent, it means that can be executed multiple times in a short period of time when the user is registering a book and need to see what are the possible categories it could belong etc)
- **Accept-Language**: en-US,en;q=0.8,q=0.5 (preference for american english)
- **User-Agent:** Mozilla/5.0 (Windows NT 10.0; Win64;x64) - or any other data about the requester device.
- **Accept-Encoding:** gzip, deflate, br (content encoding which client can understand)
- **Referer:** https://library.com
- **Authorization: "**Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjE2NDg4ODk3MzUsInVzZXJfbmFtZSI6ImFuZHJlc0BoYXNoLmNvIiwiYXV0aG9yaXRpZXMiOlsiUk9MRV9GQVJNRVIiXSwianRpIjoiMmVhYWNiODItMjdiZi00NTExLTg0Y2ItZTc1NmJjZDEyYTNmIiwiY2xpZW50X2lkIjoiY2xpZW50ZSIsInNjb3BlIjpbInJlYWQiLCJ3cml0ZSJdfQ.CHpvjSmGrbp-O1ebAmMVLqMs2H0XO7QgvZNuFppl6zE" (this token can belong to a administrator or a customer)

Response:

- **HTTP STATUS CODE 200 OK:**

  Response Body:

  The response object in this case represents the state of the categories queried by the user. The status 200 means that the resource (in this case a list of categories) has been retrieved successfully.

  JSON Representation:


```
[{
   "id":1,
   "name":"Biography",
   "description":" form of literature, commonly considered nonfictional, the subject of which is the life of an individual. One of the oldest forms of literary expression, it seeks to re-create in words the life of a human being"
},
{
   "id":2,
   "name":"fiction",
   "description":" literature created from the imagination, not presented as fact, though it may be based on a true story or situation. Types of literature in the fiction genre include the novel, short story, and novella."
}...]
```


  XML Representation:


```
<?xml version="1.0" encoding="UTF-8" ?>
<root>
 <row>
  <id>1</id>
  <name>Biography</name>
  <description> form of literature, commonly considered nonfunctional, the subject of which is the life of an individual. One of the oldest forms of literary expression, it seeks to re-create in words the life of a human being</description>
 </row>
 <row>
  <id>2</id>
  <name>fiction</name>
```

<description> literature created from the imagination, not presented as fact, though it may be based on a true story or situation. Types of literature in the fiction genre include the novel, short story, and novella.</description>
</row>
…
</root>

Response Headers:

server: nginx

date: Wed, 22 Feb 2023 19:31:16 GMT

Content-Type: application/json

Connection: keep-alive (keep the connection alive for next request)

Cache-Control: max-age=1800 (This request will be cached because is recurrent, it means that can be executed multiple times in a short period of time when the user is registering a book and need to see what are the possible categories it could belong etc)

- **HTTP STATUS CODE 401 UNAUTHORIZED**

  This code is returned when the token provided in the authorization header has expired or is not valid to carry out the request. The specific error and error code is described in the response body, for instance:

  JSON Representation:

  ```
  {
  "errorCode": "ER1",
  "error": "You are not authorized. Insufficient Permissions"
  }
  ```

  XML Representation:

  ```
  <?xml version="1.0" encoding="UTF-8" ?>
  <root>
   <errorCode>ER1</errorCode>
   <error>You are not authorized. Insufficient Permissions</error>
  </root>
  ```

- **HTTP STATUS CODE 404 NOT FOUND:**

This code is returned when the category with category_id doesn’t exist.