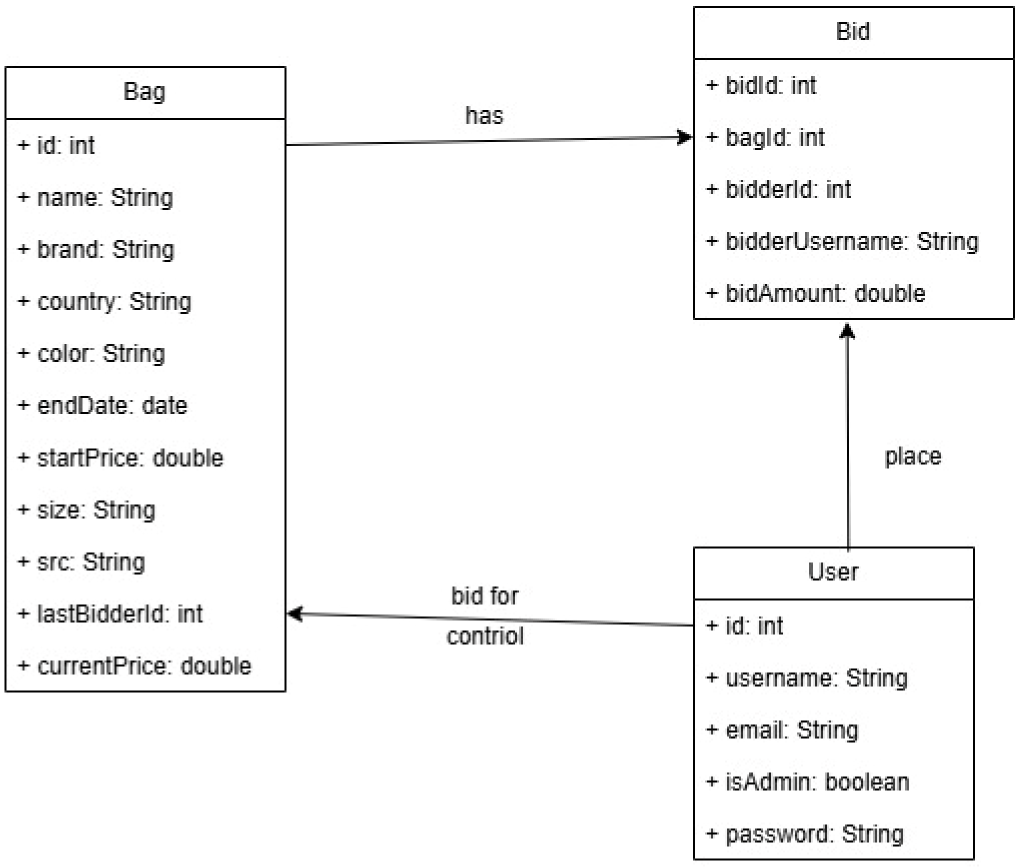
API specification file

Table of Contents

1. Class diagram



The system comprises three main classes: User, Bag, and Bid.

1. User: This class represents the users of the website. Each user has a unique id and username. The email and password are also stored in this class. A Boolean value is used to identify if this user is an admin staff or a client.

2. Bag: This class encapsulates the auction item. Each bag has a unique id, along with attributes for its name, brand, country, color, end date, start price, size, the latest bidder and current price. Only the admin users are authorized to upload, modify, and delete bags.

3. Bid: This class is associated with the Bag class. The attributes of this class include a unique bid_id (automatically generated by the application), bagId(which item is bidded for), bidAmount(how much is offered), bidderId and bidderUsername(who placed this bidder). Only the non-admin users are allowed to place bids. The bids can only be added but not be modified or deleted.

In summary, the User class manages user identities and role, the bag class handles bag uploads and management, the bid class manages bids placed for each bag.

2. GET requests

GET	**/bags?color=value&brand=value&size=value**		
Get all bags with or without being filtered from backend			
Parameters:	**Name**	**Type**	**Description**
*Add a * to the name of required parameters.*	color	query	Filters on color
	brand	query	Filters on brand
	size	query	Filters on size
Responses:	**Code**	**Description / example if successful**	
	200	All filtered bags are returned successfully.	
	200	No bags are found after the filter, an empty Json is returned.	

GET	**/bags/{bagId}**		
Get one bag with the specified id			
Parameters:	**Name**	**Type**	**Description**
*Add a * to the name of required parameters.*	id	path	Select on bag id
Responses:	**Code**	**Description / example if successful**	
	200	The bag is found and returned.	
	404	The bag is not found.	

GET	**/bags/{bagId}/bids**		
Get all bids belong to one specific bag based on its specific id			
Parameters:	**Name**	**Type**	**Description**
*Add a * to the name of required parameters.*	id	path	Select on bag id
Responses:	**Code**	**Description / example if successful**	
	200	The filtered bids are returned successfully.	

3. POST requests

POST	/bags		
Adds a new bag			
Parameters:	**Name**	**Type**	**Description**
*Add a * to the name of required parameters.*			
Responses:	**Code**	**Description / example if successful**	
	201	If new bag is created successfully. Body will contain the newly created bag information in Json form.	
	400	The information of the new added bag is not completed, or not in a valid format.	
	406	The request is not acceptable, including the start price is not a positive number or end date is a past date.	

POST	/bags/{bagId}/bids		
Add a new bid to one bag based on its specific bag id.			
Parameters:	**Name**	**Type**	**Description**
*Add a * to the name of required parameters.*	bagId	path	Select on bag id
Responses:	**Code**	**Description / example if successful**	
	201	New bid is successfully created. Body will contain the newly created place. Either plain or JSON, based on the form used in the request.	
	404	The bag is not found.	
	403	The request is not operated by authorized user.	
	406	The request is not acceptable (ex: the bid amount is smaller than the start price or not a valid number)	
	400	The auction of this bad has already ended.	

4. PUT requests

PUT	/bags/{bagId}		
Change the data of one bag based on its specific bag id			
Parameters:	**Name**	**Type**	**Description**
*Add a * to the name of required parameters.*	bagId	path	Select on bag id
Responses:	**Code**	**Description / example if successful**	
	200	Update is successful. Body will contain the newly created place. Either plain or JSON, based on the form used in the request.	
	404	The bag is not found.	
	406	The request is not acceptable, including the start price is not a positive number or end date is a past date.	

5. DELETE requests

DELETE	/bags/{bagId}		
Delete one bag based on its specific bag id.			
Parameters:	**Name**	**Type**	**Description**
*Add a * to the name of required parameters.*	bagId	path	Select on bag id
Responses:	**Code**	**Description / example if successful**	
	200	The bag is deleted.	
	404	The bag to be deleted is not found.	