

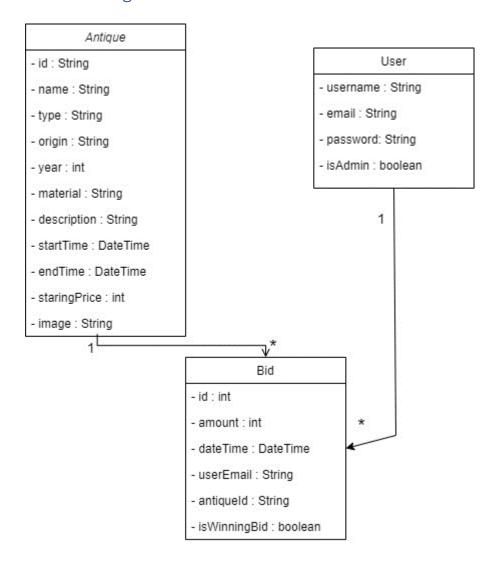
DHI2V.So
DIMITAR PEVETSOV
WEB ADVANCED

API SPECIFICATION

Table of Contents

Class diagram	2
-	
·	
•	
	Class diagram GET requests POST requests PATCH requests DELETE requests

1. Class diagram



2. GET requests

GET /ar	ntiques		
Gets all the ar	-		
Parameters:	Name	Туре	Description
* required	year	query	Returns antiques based on the giver
			year
	Origin	query	Return antiques based on the given
			origin
	Material	query	Returns antiques based on the giver
			material
	Search	query	Returns antiques based on the giver
			search prompt
Responses:	Code		/ example if successful
	200	Got the list of	of antiques which can be empty
	ntiques/{id}		
Gets antique v	with the specific	ed id	
	T		
Parameters:	Name	Туре	Description
* required	Id*	path	Id of the antique to find
Responses:	Code	Description	/ example if successful
порешее.	200	•	for the antique with the specified id
	404		vith that id is not found
	1	1	
GET /us	sers		
Get all users			
Get all asers			
Parameters:	Name	Туре	Description
* required	Italiic	.,,,,	Description.
required			
Responses:	Code	Description	/ example if successful
iveshouses.	200	•	of users which can be empty
		GOT THE HST C	a dacta willen can be empty
GET /us	sers/{email}		_
		ad amail	
Gets the user	with the specifi	eu emaii	

Parameters:	Name	Type	Description
* required	email*	path	Email of user to find
Responses:	Code	Description	/ example if successful
	200	C = 4 4 1 = 4 = 4	a fautha
	200	Got the dat	a for the user with the specified email

GET /	antiques/{id}/b	ids		
Gets all bids for the antique with the specified id				
Parameters:	Name	Type	Description	
	Id*	path	Id of antique to find	
* required				
Posnonsos	Code	Doscription	/ example if successful	
Responses:		-	•	
	200	Got the list	of bids which can be empty	
	404	No bids fou	nd for that antique	

GET /u	sers/{email}/bids				
Gets the bids for the user with the specified email					
	1	1_	12		
Parameters:	Name	Туре	Description		
* required	Email*	path	Email of user to find		
Responses:	Code	Description	Description / example if successful		
	200	Got the list	Got the list of bids which can be empty		
	404	No bids fou	No bids found for that user		

GET /u	sers/{email}/v	vinners	
Get the winni	ng bids for the	user with the spe	cified email
		<u></u>	
Parameters:	Name	Type	Description
	Email*	Path	Email of user to find
Responses:	Code	Description	/ example if successful

3. POST requests

POST	/antiques	tiques			
Adds an antique					
Parameter	s: Name	Type	Description		
* required	antique*	body	The antique being added. JSON body gets filled		
Responses	: Code	Description	Description / example if successful		
	201	Antique cre	Antique created successfully and JSON body gets filled		
	400	Antique cou or invalid	Antique could not be created, because fields are empty		

POST	/us	ers			
Adds a us	er				
Paramete	ers:	Name	Туре	Description	
* required	1	user*	body	The user that is being added. JSON body gets filled	
Pasnansa	·c•	Code	Doscription	/ example if successful	
Response	3.	201	•	d successfully and JSON body gets filled	
		400		User could not be created, because fields are empty or	
		500		User could not be created, because there is an error with the hashing of password or JWT creation.	

POST	/antiques/{id}/b	oids			
Creares a bid for an antique with the specified id					
Parameter	rs: Name	Туре	Description		
* required	bid*	body	The bid that is being created. JSON body gets gilled.		
	Id*	Path	Id of antique to find		
		I			
Responses	: Code	Description	/ example if successful		
	201 Bid created successfully and JSON body gets f		successfully and JSON body gets filled		
	400	Bid could no invalid	Bid could not be created, because fields are empty or invalid		

POST	/tokens
Creating a	a token for logging in.

Parameters:	Name	Туре	Description
	Token*	Body	The token that is being created
	-	T	
Responses:	Code	Description / example if successful	
	201	Token succe	essfully created.
	400	Token was r	not created as the email and password in
		the body wa	as not found or invalid
	500	Token was r	not created as there was an error creating
		the JWT or comparing the password from the body	

4. PATCH requests

PATCH ar	ntiques/{id}				
Modifies an antique, based on the given id					
Parameters:	Name	Туре	Description		
* required	id *	path	The id of the antique to be modified		
	antique *	body	The antique body to modify		
Responses:	Code	Description	Description / example if successful		
	200	Body contai	Body contains the modified antique.		
	400	Body was no	Body was not modified, because a field is invalid or		
		empty wher	n trying to modify or body id does not		
		match the p	match the path id		

5. DELETE requests

DELETE	/antiques/{id}	ntiques/{id}				
Deletes the antique with the specified id						
Parameters	s: Name	Туре	Description			
* required	id*	path	Id of the antique to delete			
Responses:	Code	Description	Description / example if successful			
	200	Antique del	Antique deleted			
	404	A antique w	A antique with that Id is not found			

DELETE /ar	ntiques/{id}/bids/{id}				
Deletes data f	or bid with the	e specified id			
Parameters:	Name	Туре	Description		
* required	id*	path	Id of the bid to delete		
	Id*	path	Id of the antique the bid belongs to		
Responses:	Code	Description	Description / example if successful		
	200	Bid deleted	Bid deleted		
	404	A bid or ant	A bid or antique with that Id is not found		
	400	Bid was not	Bid was not deleted as auction has ended		