API DOCUMENTATION

A web API which purpose is to function as a tool for selling and buying items.

POST /api/users/register

Description: register a new user to the system

Request body parameters:

name	type	description	required
_id	ObjectId	User's auto assigned ID	false
username	String	User's username	true
password	String	User's password	true
firstName	String	User's first name	true
lastName	String	User's last name	true
phoneNumber	String	User's phone number	true
email	String	User's email	true
date	Date	User's register date	false

Request body example:

```
{
        “username”: “testUser”,
        “password”: “testPass”,
        “firstName”: “test”,
        “lastName”: “name”,
        “phoneNumber”: “0501231234”,
        “email”: “test.email@email.com”
}
```

Responses:

code	status	success
200	User created successfully	true
400	Errors in form	false

Response body example:

```
{
        “user”: {
                “_id”: “5e50eed25169c02f68253a49”,
                “username”: “testUser”,
                “password”: “hashedPass”,
                “firstName”: “test”,
                “lastName”: “name”,
                “phoneNumber”: “0501231234”,
                “email”: “test.email@email.com”,
                “date”: “2020-02-22T00:00:00.000Z”
        }
        “status”: “User created successfully”,
        “success”: true
}
```

POST /api/users/login

Description: login with an existing user

Request body parameters:

name	type	description	required
username	String	User’s username	true
password	String	User’s password	true

Request body example:

```
{
        “username”: “testUser”,
        “password”: “testPass”,
}
```

Responses:

code	status	success
200	Token created	true
400	User not found	false
400	Password incorrect	false

Response body example:

```
{
        "success": true,
        "status": "Token created",
        "token":"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjVlNTBlZWQyNTE2OWMwMmY2ODI1M2E3O
        CIsInVzZXJuYW1lIjoib21pc3RhamEiLCJpYXQiOjE1ODIzNjY1NTAsImV4cCI6MTU4MjM2NzU1MH0.SbsEB
        O__mP5lard7RK5jM32lo_IclVvf8fHohM4PUUQ"
}
```

POST /api/items/add

Description: add a new item to the system, can’t add images with this route

Request body parameters:

name	**type**	**description**	**required**
_id	ObjectId	Item’s auto assigned ID	false
title	String	Item’s title	true
description	String	Description of the item	true
category	String	Category of the item	true
location	Object	Object containing the item’s country and city	true
img	Array	Array of objects containing the photo’s information	true
price	Float	Price of the item	true
date	Date	Item’s posting date	false
deliveryType	String	Item’s delivery type	true
sellerInfo	ObjectId	Item’s seller’s ID	false

Request body example:

```
{
        “title”: “testItem”,
        “description”: “test item to be submitted”,
        “category”: “sports”,
        “location”: {
                “country”: “Finland”,
                “city”: “Oulu”
        },
        “price”: “64.19”,
        “deliveryType”: “pickup”
}
```

Responses:

code	status	success
200	User created successfully	true
401	Unauthorized	false
400	Errors in form	false

Response body example:

```
{
        “item”: {
                “_id”: “5e50eed25169c02f68253o24”,
                “title”: “testItem”,
                “img”: [],
                “description”: “test item to be submitted”,
                “category”: “sports”,
                “location”: {
                        “country”: “Finland”,
                        “city”: “Oulu”
                },
                “price”: “64.19”,
                “deliveryType”: “pickup”,
                “sellerInfo”: “5e50eed25169c02f68253a49”
        },
        “status”: “Item created succesfully”,
        “success”: true
}
```

PUT /api/items/add-image/:id

Description: add images (max 4) to the specified item

Request body parameters:

name	type	description	required
img	Array	Array of image information	true

Request body example:

```
{
        “img”: “{path/to/image}”,
        “img”: “{path/to/image}”,
}
```

Responses:

code	status	success
200	Images added succesfully	true
401	Unauthorized	false
404	No such item	false

Response body example:

```
{
    “item”: {
        “_id”: “5e50eed25169c02f68253o24”,
        “title”: “testItem”,
        “img”: [
            {
                “fieldname”: “img”,
                “originalname”: “pyörätuoli.jpg”,
                “encoding”: “7bit”,
                “mimetype”: “image/jpeg”,
                “destination”: “../uploads”,
                “filename”: “01höbau04jgap”,
                “path”: “..\uploads\01höbau04jgap”,
                “size”: “138662”
            },
            {
                “fieldname”: “img”,
                “originalname”: “mönkijä.jpg”,
                “encoding”: “7bit”,
                “mimetype”: “image/jpeg”,
                “destination”: “../uploads”,
                “filename”: “04öajhsdg4thg”,
                “path”: “..\uploads\04öajhsdg4thg”,
                “size”: “666”
            },
        ],
        “description”: “test item to be submitted”,
        “category”: “sports”,
        “location”: {
            “country”: “Finland”,
            “city”: “Oulu”
        },
        “price”: “64.19”,
        “deliveryType”: “pickup”,
        “sellerInfo”: “5e50eed25169c02f68253a49”
    },
    “status”: “Images added succesfully”,
    “success”: true
}
```

PUT /api/items/edit/:id

Description: edit an existing item

Request body parameters:

name	type	description	required
_id	ObjectId	Item’s auto assigned ID	false
title	String	Item’s title	true
description	String	Description of the item	true
category	String	Category of the item	true
location	Object	Object containing the item’s country and city	true
img	Array	Array of objects containing the photo’s information	true
price	Float	Price of the item	true
date	Date	Item’s posting date	false
deliveryType	String	Item’s delivery type	true
sellerInfo	ObjectId	Item’s seller’s ID	false

Request body example:

```
{
        “title”: “testItem”,
        “description”: “test item to be submitted”,
        “category”: “sports”,
        “location”: {
                “country”: “Finland”,
                “city”: “Oulu”
        },
        “price”: “64.19”,
        “deliveryType”: “pickup”
}
```

Responses:

code	status	success
200	User created successfully	true
404	No such item or unauthorized	false
400	Some fields are empty	false

Response body example:

```
{
        “item”: {
                “_id”: “5e50eed25169c02f68253o24”,
                “title”: “testItem”,
                “img”: [],
                “description”: “test item to be submitted”,
                “category”: “sports”,
                “location”: {
                        “country”: “Finland”,
                        “city”: “Oulu”
                },
                “price”: “64.19”,
                “deliveryType”: “pickup”,
                “sellerInfo”: “5e50eed25169c02f68253a49”
        },
        “status”: “Item updated succesfully”,
        “success”: true
}
```

DELETE /api/items/delete/:id

Description: delete an existing item

Responses:

code	**status**	**success**
200	Deletion successful	true
404	No such item or unauthorized	false

Response body example:

```
{
        “status”: “Deletion successful”,
        “success”: true
}
```

GET /api/items/get

Description: get all items in the db

Responses:

code	status	success
200	Items found	true
404	Items not found	false

Response body example:

{

items,

"status": "Items found",

"success": true

}

GET /api/items/get/category/:category

Description: get items based on their category

Responses:

code	status	success
200	Items found	true
404	Items not found	false

Response body example:

{

items,

"status": "Items found",

"success": true

}

GET /api/items/get/location/:country/:city?

Description: get items based on their location based on required city parameter and the optional city parameter

Responses:

code	status	success
200	Items found	true
404	Items not found	false

Response body example:

```
{
        items,
        “status”: “Items found”,
        “success”: true
}
```

GET /api/items/get/date/:date

Description: get items based on their date of posting

Responses:

code	status	success
200	Items found	true
404	Items not found	false

Response body example:

```
{
        items,
        “status”: “Items found”,
        “success”: true
}
```