JB-Brunch - API document

Description	Method + Path	Request / Response

Retrieves all the users.
 	
This endpoint is used for both *login* and *register*.
	POST
	/api/allusers
		<mark>*In case a user registers (step 1 – check for ID duplicates):*</mark>
		
		Request: {
		ID: 38136514684,
		userName: "sima"
		
		}
		Response: {
		    "idDuped": false,
		    "userNameDuped": false
		}
		
		<mark>*In case a user logs in:*</mark>
		Request:{
		userName: "julian",
		password: "awdawd"
		}
		Response: {
		    "address": {
		        "city": "Rishon LeZion",
		        "street": "Ravid",
		        "houseNum": 4,
		        "apt": 6
		    },
		    "_id": "5ca4acf1eec7292434c5e9e5",
		    "firstName": "Julian",
		    "lastName": "Shikui",
		    "userName": "julian",
		    "email": juliandave1989@gmail.com",
		    "userIdNum": 201016516,
		    "password": "awdawd",
		    "role": "user",
		    "__v": 0
		}

Logs out of the system by destroying the active user session.	**POST**
	/api/register
		Response: [
		    {
		        "msg": "Session destroyed"
		    }
		]

Retrieves all products.	**GET**
	/api/products
		Response:
		[
		    {
		        "_id":
		"5cc85dd947daea27ac32f1de",
		        "title": "Brown eggs",
		        "type":
		"5cab705b240e0a1b90211b09",
		        "description": "Raw organic brown eggs in a basket",
		        "url": "uploads/0.jpg",
		        "price": 28.1
		    },...]

Creates a new user.	POST /api/register	*Request*:{
		firstName: "Dima",
		lastName: "Develovich",
		city: "Rishon Le Zion",
		street: "Ha'Nevel",
		houseNumber: 11,
		aptNumber: 1,
		userName: "dimdim5000",
		email: "dimadev@gmail.com",
		ID: 2523423451,
		choosePassword: "dima123"
		}
		
		Response: {
		    "_id": "5ccb0725c236352aec6d22f0",
		    "firstName": "Dima",
		    "lastName": "Develovich",
		    "address": {
		        "city": "Rishon Le Zion",
		        "street": "Ha'Nevel",
		        "houseNum": 11,
		        "apt": 1
		    },
		    "userName": "dimdim5000",
		    "email": "dimadev@gmail.com",
		    "userIdNum": 2523423451,
		    "password": "dima123",
		    "role": "user",
		    "__v": 0
		}

Retrieves all the product categories.	GET /api/category	*Response*:
		[
		    {
		        "_id":
		"5cab6fdbf0eb9b1b90c11de3",
		        "category": "dairy"
		    },
		    {
		        "_id":
		"5cab7008240e0a1b90211b06",
		        "category": "vegetable"
		    }...]

Creates a new cart for the user.	POST /api/createCart	*Request*:
		{
		userID: "4cab5ffaf078v1b12dhda2"
		}
		
		Response:
		{
		    "_id": "5ccb0e0ac236352aec6d22f1",
		    "creationDate": "Thu May 02 2019
		18:34:34 GMT+0300 (Israel Daylight
		Time)",
		    "__v": 0
		}

Searches a user's cart (user service).	**Get /api/searchCart**	***Request****:* *{* *userID: "5ca4acf1eec7292434c5e9e5"* *}*

Searches a user's cart (shop service).	**Get /api/searchUserCart**	***Response****:*
		{
		    "_id": "5cc9805ce85b972a2c803e92",
		    "client":
		"5ca4acf1eec7292434c5e9e5",
		    "creationDate": "Wed May 01 2019
		14:17:48 GMT+0300 (Israel Daylight
		Time)",
		    "__v": 0
		}

Inserts a product to the cart.
 	
This endpoint will initially query the selected product ID in the user's cart. It will then Assign the necessary implementation method:
 	
if the cart lacks said product: item will be added via:
 	
ƒ insertToCart(request).
 	
If the cart already includes the product: item will be updated via:
 	
ƒ updateProdStatus({ product_ID, amount, summed_Price});
	GET /api/insertToCartMethod	***Request****:{*
		Cart: "5cc9805ce85b972a2c803e92",
		chosenProductID:
		"5cc85dd947daea27ac32f1de",
		amount: 2
		totalPrice: 56.2
		}
		
		==*If cart does not contain the chosen product's ID:*==
		
		Response*:*
		{
		    "_id": "5ccb1e28f18c7f33cc54176c",
		    "prodID":
		"5cc85dd947daea27ac32f1de",
		    "cartID":
		"5cc9805ce85b972a2c803e92",
		    "amount": 2,
		    "totalPrice": 56.2,
		    "__v": 0
		}
		
		==*If cart already contains the chosen product's ID:*==
		
		Response*:*
		{
		    "_id": "5ccb1e28f18c7f33cc54176c",
		    "prodID":
		"5cc85dd947daea27ac32f1de",
		    "cartID":
		"5cc9805ce85b972a2c803e92",
		    "amount": 4,
		    "totalPrice": 112.4,
		    "__v": 0
		}

Searches the user's cart contents.	**GET** **/api/searchUserProducts**	*Request*: {
		CartID: "5cc9805ce85b972a2c803e92"
		}
		
		Response:
		[{
		    "_id":
		"5ccb1e28f18c7f33cc54176c",
		        "prodID":
		"5cc85dd947daea27ac32f1de",
		        "cartID":
		"5cc9805ce85b972a2c803e92",
		        "amount": 4,
		        "totalPrice": 112.4,
		        "__v": 0
		}...]

Retrieves each product's info by a mapped array of the user's cart product IDs.	**GET** **Api/displayUserProducts**	*Request*:
		{
		prod[0]: "5cc85dd947daea27ac32f1e1",
		prod[1]: "5cc85dd947daea27ac32f1f8",
		prod[2]: "5cc85dd947daea27ac32f1fc"
		}
		
		Response:
		{
		    "userProd": [
		        {
		            "_id":
		"5cc85dd947daea27ac32f1e1",
		            "title": "Green smoothie",
		            "type":
		"5cab6fdbf0eb9b1b90c11de3",
		            "description": "Glass of
		green smoothie with quail egg's yolk,
		with cocktail tube, apple and baby
		spinach leaves over tin surface.",
		            "url": "uploads/3.jpg",
		            "price": 17.68
		        }...]
		}

Deletes a product from the cart.
 	
The id parameter addresses the "cart products" collection
	DELETE **api/del/:id**	*Request*: {
		cartID: "5cc9805ce85b972a2c803e92"
		}
		
		Response:
		{
		    "n": 1,
		    "ok": 1
		}

Deletes the cart (will not delete the cart's contents).
 	
The id parameter addresses the *"carts"* collection
	DELETE
	api/delCart/:id
		Response:
		`{`
		`    "n": 1,`
		`    "ok": 1`
		`}`

Deletes the cart and all of its contents.
 	
The id parameter addresses both *"carts"* and *"cart products"* collection
	DELETE
	api/discardCart/:cartID
		Response:
		`{`
		`    "deletedCart": {`
		`        "n": 1,`
		`        "ok": 1`
		`    },`
		`    "deletedProds": {`
		`        "n": 2,`
		`        "ok": 1`
		`    }`
		`}`

Retrieves all orders from the current user.	**GET**
	/api/searchOrder
		Request:{
		userID: "5ca4acf1eec7292434c5e9e5"
		}
		
		Response:
		`[`
		`    {`
		`        "address": {`
		`            "city": "Rishon LeZion",`
		`            "street": "Ravid",`
		`            "houseNum": 4,`
		`            "apt": 6`
		`        },`
		`        "payment": {`
		`            "card": "3223",`
		`            "cvv": "123",`
		`            "cardExp": "Wed May 29 2019 00:00:00 GMT+0300 (Israel Daylight Time)"`
		`        },`
		`        "_id": "5cc0830da488d3093cadd39e",`
		`        "userID": "5ca4acf1eec7292434c5e9e5",`
		`        "cartID": "5cc0823b4b420a0a68f9958a",`
		`        "debit": 178.91,`
		`        "scheduled": "Thu Apr 25 2019 00:00:00 GMT+0300 (Israel Daylight Time)",`
		`        "generated": "Wed Apr 24 2019 18:38:53 GMT+0300 (Israel Daylight Time)",`
		`        "__v": 0`
		`    }...]`

| Creates a new order. | POST api/orders | Request:{
userID: ”5ca4acf1eec7292434c5e9e5”
cartID: “5cc60f6ac4405b1330fbdfb31”
debit: 410.38
city: “Rishon Le Zion”
street: “Ravid”
houseNum:4
apt:6
scheduled: “Mon May 27 2019 00:00:00 GMT+0300 (Israel Daylight Time)”
generated: “Fri May 03 2019 15:16:34 GMT+0300 (Israel Daylight Time)”
card: 3123
cvv: 136
cardExp: “Wed Feb 09 2022 00:00:00 GMT+0200 (Israel Standard Time)”
}

Response:
{
    "_id": "5ccc3311c562350478d8d498",
    "userID": "5ca4acf1eec7292434c5e9e5",
    "cartID": "5cc60f6ac4405b1330fbdfb31",
    "debit": 410.38,
    "scheduled": "Mon May 27 2019 00:00:00 GMT+0300 (Israel Daylight Time)",
"generated": "Fri May 03 2019 15:16:34 GMT+0300 (Israel Daylight Time)",
    "address": {
        "city": "Rishon Le Zion",
        "street": "Ravid",
        "houseNum": 4,
        "apt": 6

    },
  "payment": {
        "card": "3123",
        "cvv": "136",
        "cardExp": "Wed Feb 09 2022 00:00:00 GMT+0200 (Israel Standard Time)"
    },
    "__v": 0
} |

Retrieves all the orders.	**GET /api/allOrders**	*Response:*
		[
		    {
		        "address": {
		            "city": "Rishon LeZion",
		            "street": "Ravid",
		            "houseNum": 4,
		            "apt": 6
		        },
		        "payment": {
		            "card": "3223",
		            "cvv": "123",
		            "cardExp": "Wed May 29 2019 00:00:00 GMT+0300 (Israel Daylight Time)"
		        },
		        "_id": "5cc0830da488d3093cadd39e",
		        "userID": "5ca4acf1eec7292434c5e9e5",
		        "cartID": "5cc0823b4b420a0a68f9958a",
		        "debit": 178.91,
		        "scheduled": "Thu Apr 25 2019 00:00:00 GMT+0300 (Israel Daylight Time)",
		        "generated": "Wed Apr 24 2019 18:38:53 GMT+0300 (Israel Daylight Time)",
		        "__v": 0
		    }...]

Locks the cart's products.
 	
Updates all the user's cart IDs in the "cart products" collection by adding a string: "LOCKED" + *order formation date*.
 	
(This endpoint serves as a progression to the ordering process).
	PUT /api/lockCartProd	*Request*: {
		cartID: "5ccc48c8c562350478d8d49a"
		generated: "Fri May 03 2019 16:38:20 GMT+0300 (Israel Daylight Time)"
		}
		
		Response: {
		    "n": 2,
		    "nModified": 2,
		    "ok": 1
		}

Locks the cart.
 	
Updates the active cart user's ID in the "*carts*" collection by adding a string: "LOCKED" + *order formation date*.
 	
(This endpoint serves as a progression to the ordering process).
	PUT /api/lockCart	*Request*: {
		userID: "5ca4acf1eec7292434c5e9e5"
		generated: "Fri May 03 2019 16:38:20 GMT+0300 (Israel Daylight Time)"
		}
		
		Response: {
		    "n": 1,
		    "nModified": 1,
		    "ok": 1
		}

Adds a new product to the "products" collection. Admin endpoint	**POST /api/adminProduct**	*Request*: { *title: "Traditional Irish Lamb Stew"* *type: "5cab705b240e0a1b90211b09"* *description: "Authentic Irish stew is always made with lamb and is even better as the flavors blend, so make it the day before you plan to serve it".* *url: "http://www.preparednesspro.com/wp-content/uploads/2012/03/irish_stew.jpg"* *price:21* *}* *Response*: *{* *"_id": "5ccc626bc562350478d8d4a0",* *"title": "Traditional Irish Lamb Stew",* *"type": "5cab705b240e0a1b90211b09",* *"description": "Authentic Irish stew is always made with lamb and is even better as the flavors blend, so make it the day before you plan to serve it.",* *"url": "http://www.preparednesspro.com/wp-content/uploads/2012/03/irish_stew.jpg",* *"price": 21,* *"__v": 0* *}*
Retrieves a single product from the "products" collection. Admin endpoint	**GET /api/adminProduct**	*Request*: { *prodID: "5cc85dd947daea27ac32f209"* *}* *Response*: [*{* *"_id": "5cc85dd947daea27ac32f209",* *"title": "Strawberries",* *"type": "5cab7022240e0a1b90211b07",* *"description": "Healthy breakfast set. rice cereal or porridge with fresh strawberry, , and honey over white rustic wood backdrop.",* *"url": "uploads/44.jpg",* *"price": 15.13* *}* *]*

Updates a single product from the "products" collection.
 	
Admin endpoint
	PUT
	/api/adminProduct
		Request:{
		prodID:5cc85dd947daea27ac32f1e6
		title:Lemon and salt
		type:5cab7022240e0a1b90211b07
		description:Rosemary and salt on the table.
		url:uploads/9.jpg
		price:15.79
		}
		
		Response: {
		    "_id": "5cc85dd947daea27ac32f1e6",
		    "title": "Lemon and salt",
		    "type": "5cab7022240e0a1b90211b07",
		    "description": "Rosemary and salt on the table.",
		    "url": "uploads/9.jpg",
		    "price": 15.79
		}

Uploads a file (image) Admin endpoint *Sends a multipart/form-data object to the Multer middleware to handle.* *Unique time stamp is given through client side* *File type restricted to 'image/jpeg' through client side*	**POST** **/api/ admin_img_upload**	*Request: {* *productImage: Img_file* *}* *Response: {* *'file_name.jpg'* *}*
Retrieves api doc via: *f sendFile(file_path)*	**POST** **/shopping/api**	*Response: Blob object will be downloaded automatically via client side "file-saver npm"*