

# JB-Brunch - API document

Description	Method + Path	Request / Response
<p>Retrieves all the users.</p> <p>This endpoint is used for both <i>login</i> and <i>register</i>.</p>	<p>POST /api/allusers</p>	<p><b>In case a user registers (step 1 - check for ID duplicates):</b></p> <p><u>Request:</u> { ID: 38136514684, userName: "sima" }</p> <p><u>Response:</u> { "idDuped": false, "userNameDuped": false }</p> <p><b>In case a user Logs in:</b></p> <p><u>Request:</u>{ userName: "julian", password: "awdawd" }</p> <p><u>Response:</u> { "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 }</p>
<p>Logs out of the system by destroying the active user session.</p>	<p>POST /api/register</p>	<p><u>Response:</u> [ { "msg": "Session destroyed" } ]</p>
<p>Retrieves all products.</p>	<p>GET /api/products</p>	<p><u>Response:</u> [ { "_id": "5cc85dd947daea27ac32f1de", "title": "Brown eggs", "type": "5cab705b240e0a1b90211b09", "description": "Raw organic brown eggs in a basket", "url": "uploads/0.jpg", "price": 28.1 },...]</p>

Creates a new user.	POST /api/register	<p><b><u>Request:</u></b>{  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"  }</p> <p><b><u>Response:</u></b> {  "_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  }</p>
Retrieves all the product categories.	GET /api/category	<p><b><u>Response:</u></b>  [  {  "_id":  "5cab6fdbf0eb9b1b90c11de3",  "category": "dairy"  },  {  "_id":  "5cab7008240e0a1b90211b06",  "category": "vegetable"  }...]</p>
Creates a new cart for the user.	POST /api/createCart	<p><b><u>Request:</u></b>  {  userID: "4cab5ffaf078v1b12dhda2"  }</p> <p><b><u>Response:</u></b>  {  "_id": "5ccb0e0ac236352aec6d22f1",  "creationDate": "Thu May 02 2019 18:34:34 GMT+0300 (Israel Daylight Time)",  "__v": 0  }</p>

Searches a user's cart (user service).	Get /api/searchCart	<u><b>Request:</b></u> <pre>{   userID: "5ca4acf1eec7292434c5e9e5" }</pre> <u><b>Response:</b></u> <pre>{   "_id": "5cc9805ce85b972a2c803e92",   "client":     "5ca4acf1eec7292434c5e9e5",   "creationDate": "Wed May 01 2019     14:17:48 GMT+0300 (Israel Daylight     Time)",   "_v": 0 }</pre>
Searches a user's cart (shop service).  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:  <i>f insertToCart(request).</i>  If the cart already includes the product: item will be updated via:  <i>f updateProdStatus({ product_ID, amount, summed_Price});</i>	GET /api/insertToCartMethod	<u><b>Request:</b></u> <pre>{   Cart: "5cc9805ce85b972a2c803e92",   chosenProductID:     "5cc85dd947daea27ac32f1de",   amount: 2   totalPrice: 56.2 }</pre> If cart does not contain the chosen product's ID:  <u><b>Response:</b></u> <pre>{   "_id": "5ccb1e28f18c7f33cc54176c",   "prodID":     "5cc85dd947daea27ac32f1de",   "cartID":     "5cc9805ce85b972a2c803e92",   "amount": 2,   "totalPrice": 56.2,   "_v": 0 }</pre> If cart already contains the chosen product's ID:  <u><b>Response:</b></u> <pre>{   "_id": "5ccb1e28f18c7f33cc54176c",   "prodID":     "5cc85dd947daea27ac32f1de",   "cartID":     "5cc9805ce85b972a2c803e92",   "amount": 4,   "totalPrice": 112.4,   "_v": 0 }</pre>

Searches the user's cart contents.	GET /api/searchUserProducts	<p><b><u>Request:</u></b> { CartID: "5cc9805ce85b972a2c803e92" }</p> <p><b><u>Response:</u></b> [ {     "_id":     "5ccb1e28f18c7f33cc54176c",     "prodID":     "5cc85dd947daea27ac32f1de",     "cartID":     "5cc9805ce85b972a2c803e92",     "amount": 4,     "totalPrice": 112.4,     "__v": 0 } ... ]</p>
Retrieves each product's info by a mapped array of the user's cart product IDs.	GET Api/displayUserProducts	<p><b><u>Request:</u></b> { prod[0]: "5cc85dd947daea27ac32f1e1", prod[1]: "5cc85dd947daea27ac32f1f8", prod[2]: "5cc85dd947daea27ac32f1fc" }</p> <p><b><u>Response:</u></b> {     "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         } ... ]     } }</p>
<p>Deletes a product from the cart.</p> <p>The id parameter addresses the "cart products" collection</p>	DELETE api/del/:id	<p><b><u>Request:</u></b> { cartID: "5cc9805ce85b972a2c803e92" }</p> <p><b><u>Response:</u></b> {     "n": 1,     "ok": 1 }</p>

<p>Deletes the cart (will not delete the cart's contents).</p> <p>The id parameter addresses the "carts" collection</p>	<p><b>DELETE</b> api/delCart/:id</p>	<p><b><u>Response:</u></b></p> <pre>{   "n": 1,   "ok": 1 }</pre>
<p>Deletes the cart and all of its contents.</p> <p>The id parameter addresses both "carts" and "cart products" collection</p>	<p><b>DELETE</b> api/discardCart/:cartID</p>	<p><b><u>Response:</u></b></p> <pre>{   "deletedCart": {     "n": 1,     "ok": 1   },   "deletedProds": {     "n": 2,     "ok": 1   } }</pre>
<p>Retrieves all orders from the current user.</p>	<p><b>GET</b> /api/searchOrder</p>	<p><b><u>Request:</u></b>{ userID: "5ca4acf1eec7292434c5e9e5" }</p> <p><b><u>Response:</u></b></p> <pre>[   {     "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   }...]</pre>

Creates a new order.	POST api/orders	<p><b><u>Request:</u></b>{  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)"  }</p> <p><b><u>Response:</u></b>  {    "_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  }</p>
----------------------	--------------------	---

Retrieves all the orders.	GET /api/allOrders	<p><b><u>Response:</u></b></p> <pre>[   {     "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   }...]</pre>
<p>Locks the cart's products.</p> <p>Updates all the user's cart IDs in the "cart products" collection by adding a string: "LOCKED" + order formation date.</p> <p>(This endpoint serves as a progression to the ordering process).</p>	PUT /api/lockCartProd	<p><b><u>Request:</u></b> {  cartID: "5ccc48c8c562350478d8d49a"  generated: "Fri May 03 2019 16:38:20  GMT+0300 (Israel Daylight Time)"  }</p> <p><b><u>Response:</u></b> {  "n": 2,  "nModified": 2,  "ok": 1  }</p>
<p>Locks the cart.</p> <p>Updates the active cart user's ID in the "carts" collection by adding a string: "LOCKED" + order formation date.</p> <p>(This endpoint serves as a progression to the ordering process).</p>	PUT /api/lockCart	<p><b><u>Request:</u></b> {  userID: "5ca4acf1eec7292434c5e9e5"  generated: "Fri May 03 2019 16:38:20  GMT+0300 (Israel Daylight Time)"  }</p> <p><b><u>Response:</u></b> {  "n": 1,  "nModified": 1,  "ok": 1  }</p>

<p>Adds a new product to the "products" collection.</p> <p>Admin endpoint</p>	<p>POST /api/adminProduct</p>	<p><u>Request:</u> {  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  }</p> <p><u>Response:</u>  {  " _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  }</p>
<p>Retrieves a single product from the "products" collection.</p> <p>Admin endpoint</p>	<p>GET /api/adminProduct</p>	<p><u>Request:</u> {  prodID: "5cc85dd947daea27ac32f209"  }</p> <p><u>Response:</u> [  {  " _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  }  ]</p>



<p>Updates a single product from the “products” collection.</p> <p>Admin endpoint</p>	<p>PUT /api/adminProduct</p>	<p><b><u>Request:</u></b>{  prodID:5cc85dd947daea27ac32f1e6  title:Lemon and salt  type:5cab7022240e0a1b90211b07  description:Rosemary and salt on the table.  url:uploads/9.jpg  price:15.79  }</p> <p><b><u>Response:</u></b> {  “_id”: “5cc85dd947daea27ac32f1e6”,  “title”: “Lemon and salt”,  “type”: “5cab7022240e0a1b90211b07”,  “description”: “Rosemary and salt on the table.”,  “url”: “uploads/9.jpg”,  “price”: 15.79  }</p>
<p>Retrieves api doc via:  <i>f sendFile( file_path)</i></p>	<p>POST /shopping/api</p>	<p><b><u>Response:</u></b> Blob object will be downloaded automatically via client side “file-saver npm”</p>