degaso API-Documentation

Introduction

This documentaion contains all the neccessary information about how the Degaso Cash System works and how to access its backend implementation in order to interfere with the system as an administrator or a staff device.

Every Degaso Cash System has one main device, which handles all kinds of client side requests (just like sending new orders or getting information about the available products e.g. price, name etc.) and therefore also operates as a server running inside the restaurants network, which is accessable through the main devices IP Adress.

All devices that are in the same network with the main cash device, can access the main device's API and communicate with the other devices by means of a WebSocket connection to the main device. The next section comprises the neccessary information about how to connect to and get messages over the WebSocket communication channel with the main device. Since each device in the cash system must be synchronized according to the activity of other devices, this WebSocket connection step is inevitable for every device in the system.

Following the WebSocket connection, the API documentation of the main devices server, will take up the largest part in this documentation. It handles the entirety of the main devices server side implementation. This API can process all kinds of requests, from ordering a pizza for a specified table through changing the price of certain products up to getting a list of all the products the restaurant has to offer. It can be seen as the central processing unit of the cash system, whereas each connected device has to obey the rules set by this API and listen for changes made by other devices.

Websocket

To handle the communication between devices connected to the cash system, a webSocket based communication is used. The communication center is always the main device. Hence a

communication channel never takes place between two connected devices, in fact, every device will talk to the main device, which subsequently will broadcast any new information to all connected devices.

Example: A staff member goes to table 3 and orders three Pizzas. The staffs device will send a post request called '/newOrder' to the server running on the main device, processing this information. When the new order is saved on the main device, the server will broadcast the message: 'orderUpdate' to all connected devices. As a result of that, all connected devices should send a request called '/getAllActiveOrders' to get an update about all the active orders.

The WebSocket server will only send a few distinct messages, each connected device should listen to these messages:

'orderUpdate': This message will be send when an order is changed. (/newOrder, /addToOrder APIs for example)

'productUpdate': This message will be send when a product is changed. (/newProduct, /deleteProduct APIs for example)

'employeesUpdate': This message will be send when a new employee is added.

Example how to connect to the main device via a WebSocket connection and to listen to new messages with nodejs:

```
//create a WebSocket object
WebSocket = require('ws').WebSocket
ws = new WebSocket("ws:" + device_IP_Adress + ":3434");

ws.on('open', function open() {
    console.log('WebSocket opened')
});

ws.on('close', function close() {
    console.log('WebSocket opened')
});

ws.on('message', function message(data) {
    console.log(data.toString())
});
```

Please notice, that this code won't automatically reconnect to the server when the connection is lost, due to poor wifi connection or other reasons. A more comprehensive implementation would have to include a reconection timeout and a logic to listen for the specific messages listet above, to be able to react to changes made by other devices!

Rest API:

Category

Category | Change category order (Admin)

0.1.0

API to change the existing category order. After successfully changing the category order, the server will broadcast the 'productUpdate' message to all connected devices.

POST

```
device_IP_Adress:3434/changeCategoryOrder
```

```fetch
//this example will update the order value of the categorys Pasta and Pizza
fetch('http://device_IP_Adress:3434/changeCategoryOrder/', {
    method: 'POST',
    headers: {
        'Content-Type': 'application/json',
    },
    body: JSON.stringify({
        categorys:
            [{
                "name": "Pasta",
                "color": "#eee",
                "order": 0,
                "_id": "6BM5icO02iNZ9UHM"
            },
            {
                "name": "Pizza",
                "color": "#eee",
                "order": 1,
                "_id": "EC4PRqOb9Qal3Et0"
            }]
    }),
})
    .then(response => console.log(response.status))
    .catch((error) => {
        console.error('Error:', error);
    });
```

Header

Feld	Beschreibung
Content-Type	application/json

Request Body

Feld	Typ	Beschreibung

categorys	**array**	This array can hold multiple category objects. The server will iterate over each object an pick out the order value from each category and update these values. Note that the category objects already have to exist in the system. There for it is recommended to first fetch all categorys with /getAllCategorys and then update the order values according to the desired values and the send it back to the server with the /changeCategoryOrder API.
categorys._id	**string**	Unique identifier string assigned to each category
categorys.name	**string**	category name
categorys.color	**string**	unused
categorys.order	**string**	updated order value to set inside the cash system

200

Feld	Typ	Beschreibung
Statuscode	**none**	The API will return status code 200 if the request was successfull

Category | Get all Categorys (All users)

0.1.0

API to get all the categorys for the products the restaurant has to offer.

GET

```
device_IP_Adress:3434/getAllCategorys
```

```
fetch('http://device_IP_Adress:3434/getAllCategorys/', {
   method: 'GET',
   headers: {
     'Content-Type': 'application/json',
   }
   })
   .then(response => response.json())
   .then(data => {
     if (data != null) {
        console.log(data)
     }
   })
   .catch((error) => {
     console.error('Error:', error);
   });
```

Header

Feld	Beschreibung
Content-Type	application/json

200

Feld	Typ	Beschreibung
category	**array**	Returned Array with all categorys as distinc objects
_id	**string**	Unique identifier string assigned to each category
name	**string**	category name
color	**string**	unused
order	**string**	determines in which the category names are displayed in the frontend (lower first, can be changed by /changeCategoryOrder)

```
[
    {
        "name": "Pasta",
        "color": "#eee",
        "order": 1,
        "_id": "6BM5icO02iNZ9UHM"
    },
    {
        "name": "Pizza",
        "color": "#eee",
        "order": 0,
        "_id": "EC4PRqOb9Qal3Et0"
    },
    {
        "name": "Burger",
        "color": "#eee",
        "order": 2,
        "_id": "KM3WWDRsxUd3qxmX"
    }
]
```

Category | Get all Products (Admin)

0.1.0

API to delete an existing Category. After successfully deleting a category, the server will broadcast the 'productUpdate' message to all connected devices.

POST

```
device_IP_Adress:3434/deleteCategory
```

```
fetch('http://device_IP_Adress:3434/deleteCategory/',{
    method: 'POST',
    headers: {
      'Content-Type': 'application/json',
    },
    body: JSON.stringify({
      _id: 'S1BFg9NVwdU70NM6'
    }),
    }
    .then(response => console.log(response.status))
    .catch((error) => {
    console.error('Error:', error);
    });
```

Header

Feld	Beschreibung
Content-Type	application/json

Request Body

Feld	Typ	Beschreibung
_id	**string**	This value holds the _id of the category to be deleted. Only if the id value is existing, the category will be removed.

200

Feld	Typ	Beschreibung
Statuscode	**none**	The API will return status code 200 if the request was successfull

Category | Get all Products (Admin)

0.1.0

API to create a new Category. Products can then be assigned to this new category. After successfully adding a new category, the server will broadcast the 'productUpdate' message to all connected devices.

POST

```
device_IP_Adress:3434/newCategory
```

```
fetch('http://device_IP_Adress:3434/newCategory/',{
    method: 'POST',
```

```
    headers: {
      'Content-Type': 'application/json',
    },
    body: JSON.stringify({
      name: 'Pizza'
    }),
    }
    .then(response => response.json())
    .then(data => {
        console.log('Category with the following ID was added:', data);
    })
    .catch((error) => {
        console.error('Error:', error);
    });
```

Header

Feld	Beschreibung
Content-Type	application/json

Request Body

Feld	Typ	Beschreibung
name	**string**	This value holds the name of the new Category as a string. This value should not be an empty string, because it will the be rendered as a blank category in the frontend View.

200

Feld	Typ	Beschreibung
_id	**string**	If successfull, the API returns a string with the _id property of the new added category

- Example server response

```
"S1BFg9NVwdU70NM6" //
```

Configuration

Configuration | Get Employees(All users)

0.1.0

API to get the employees object from the cash system config file. The server will answer with an array containing one object, which holds the employees names in the employees property

POST

```
device_IP_Adress:3434/getEmployees
```

```
fetch('http://device_IP_Adress:3434/getEmployees/', {
    method: 'POST',
    headers: {
    'Content-Type': 'application/json',
    }
  .then(response => response.json())
    .then(data => console.log(data))
    .catch((error) => {
        console.error('Error:', error);
    });
```

Header

Feld	Beschreibung
Content-Type	application/json

```
[
    {
        "configName": "employees",
        "employees": [
            "Frederik",
            "Valentin",
            "Jan",
            "Claudio"
        ],
        "_id": "bULgDjsDMl2l778w"
    }
]
```

Orders

Orders | Add articles to order (All users)

0.1.0

This API is used to add articles to an order. After the articles were added successfully , the server will send the 'orderUpdate' message to all connected devices through the WebSocket channel.

POST

```
device_IP_Adress:3434/addToOrder
```

```
fetch('http://device_IP_Adress:3434/addToOrder/', {
method: 'POST',
headers: {
    'Content-Type': 'application/json',
},
body: JSON.stringify({
    orderId: "37J6f43wKcfd3qqw",
    table: "2",
    articles: [
        {
            name: 'Cheeseburger',
            price: '24.5',
            category: 'Burger',
            tax: '19',
            kitchenRelevant: true,
            active: true,
            combinationProduct: false,
            infoText: '',
            stockChecking: false,
            stockAmount: 0,
            order: 0,
            customPrinterAddress: 'barPrinter',
            _id: 'Y0xzNAheX7OeQ703',
            identifyForList: '864e2fbc-515a-4833-ac76-c8c39e35e8cc',
            additionalInfo: '| mit Fries| mit Potato Wedges',
            course: 0,
            brangToTable: false,
            combinableWith: [
                {
                    name: 'Fries',
                    price: '4.5',
                    active: true,
                    availableProducts: [
                        {
                            productId: 'Y0xzNAheX7OeQ703',
                            productName: 'Cheeseburger'
                        },
                        { productId: 'ZebYTu5kVZpk7jpO', productName: 'BBQ
                    ],
                    combinationProduct: true,
                    _id: 'OoeIm5jDfzoA5ILm'
                },
```

```
                {
                    name: 'Potato Wedges',
                    price: '5',
                    active: true,
                    availableProducts: [
                        { productId: 'ZebYTu5kVZpk7jpO', productName: 'BBQ
                        {
                            productId: 'Y0xzNAheX7OeQ703',
                            productName: 'Cheeseburger'
                        }
                    ],
                    combinationProduct: true,
                    _id: 'fbv0XiGlwfNmyQQn'
                },
                {
                    name: 'Chicken Wings',
                    price: '4.5',
                    active: true,
                    availableProducts: [
                        { productId: 'ZebYTu5kVZpk7jpO', productName: 'BBQ
                        {
                            productId: 'Y0xzNAheX7OeQ703',
                            productName: 'Cheeseburger'
                        }
                    ],
                    combinationProduct: true,
                    _id: 'tB8sw3h9CA05xktr'
                }
            ],
            combinedWith: [
                { productName: 'Fries', price: '4.5' },
                { productName: 'Potato Wedges', price: '5' }
            ]
        },
        {
            name: 'Margherita',
            price: '9.2',
            category: 'Pizza',
            tax: '19',
            kitchenRelevant: true,
            active: true,
            combinationProduct: false,
            infoText: '',
            stockChecking: false,
            stockAmount: 0,
            order: 0,
            customPrinterAddress: 'barPrinter',
```

```
            _id: '1ymAPOHWTFft5wFp',
            identifyForList: '3b38d0e3-3c4d-4396-86c7-9b6236570bb2',
            additionalInfo: '',
            course: 0,
            brangToTable: false,
            combinableWith: [],
            combinedWith: []
        }
    ]
},
})
.then(response => response.json())
.then(data => console.dir(data))
.catch((error) => {
    console.error('Error:', error);
});
```

Header

Feld	Beschreibung
Content-Type	application/json

Request Body

Feld	Typ	Beschreibung
orderId	**string**	This string selects the order (by its ID) from which an article should be removed.
orderId	**string**	This string selects the order (by its ID) from which an article should be removed.
table	**array**	This string holds the table number of the add to order request

200

Feld	Typ	Beschreibung
success	**none**	none

Orders | Add new order (All users)

0.1.0

API to add a new order. After the order was added, the server will send the 'orderUpdate' message to all connected devices through the WebSocket channel. On top of that, the server will go through all ordered articles and will send a print requests to the kitchen printer or the bar printer (depends on the kitchenRelevant variable of the ordered articles), so that the employees

in the kitchen or at the bar get informed by the new order and can prepare the drinks or the dishes. Furthermore the server will add some information about the order (time, paid products etc. → please refere to the /getAllActiveOrders API to get more details about the order object). In the case of this API, the user only has to send two variables which are explained below, the rest will be done by the server. Please note, that it is very important to change the stockAmount of every product that is ordered, when the stockChecking boolean is set to true. (For more information and how to properly implement this behaviour, please contact us).

To make sure not to get confused with the terminology, the diffrence between a product and an article has to be explained at this point. A product becomes an article, when it is ordered. The user has to add some additional properties to the product in order to become an article. The additionalInfo variable for example, has to be added to the product (which can hold individual customer wishes, like 'medium' in case of a burger).

POST

```
device_IP_Adress:3434/newOrder
```

```
fetch('http://device_IP_Adress:3434/newOrder/', {
    method: 'POST',
    headers: {
        'Content-Type': 'application/json',
    },
    body: JSON.stringify({
        table: '2',
        articles: [
            {
                name: 'Cheeseburger',
                price: '24.5',
                category: 'Burger',
                tax: '19',
                kitchenRelevant: true,
                active: true,
                combinationProduct: false,
                infoText: '',
                stockChecking: false,
                stockAmount: 0,
                order: 0,
                customPrinterAddress: 'barPrinter',
                _id: 'Y0xzNAheX7OeQ703',
                identifyForList: '864e2fbc-515a-4833-ac76-c8c39e35e8cc',
                additionalInfo: '| mit Fries| mit Potato Wedges',
                course: 0,
                brangToTable: false,
                combinableWith: [
                    {
                        name: 'Fries',
                        price: '4.5',
```

```
                    active: true,
                    availableProducts: [
                        {
                            productId: 'Y0xzNAheX7OeQ703',
                            productName: 'Cheeseburger'
                        },
                        { productId: 'ZebYTu5kVZpk7jpO', productName: '
                    ],
                    combinationProduct: true,
                    _id: 'OoeIm5jDfzoA5ILm'
                },
                {
                    name: 'Potato Wedges',
                    price: '5',
                    active: true,
                    availableProducts: [
                        { productId: 'ZebYTu5kVZpk7jpO', productName: '
                        {
                            productId: 'Y0xzNAheX7OeQ703',
                            productName: 'Cheeseburger'
                        }
                    ],
                    combinationProduct: true,
                    _id: 'fbv0XiGlwfNmyQQn'
                },
                {
                    name: 'Chicken Wings',
                    price: '4.5',
                    active: true,
                    availableProducts: [
                        { productId: 'ZebYTu5kVZpk7jpO', productName: '
                        {
                            productId: 'Y0xzNAheX7OeQ703',
                            productName: 'Cheeseburger'
                        }
                    ],
                    combinationProduct: true,
                    _id: 'tB8sw3h9CA05xktr'
                }
            ],
            combinedWith: [
                { productName: 'Fries', price: '4.5' },
                { productName: 'Potato Wedges', price: '5' }
            ]
        },
        {
            name: 'BBQ Burger',
```

```
                price: '21',
                category: 'Burger',
                tax: '19',
                kitchenRelevant: true,
                active: true,
                combinationProduct: false,
                infoText: '',
                stockChecking: false,
                stockAmount: 0,
                order: 0,
                customPrinterAddress: 'barPrinter',
                _id: 'ZebYTu5kVZpk7jpO',
                identifyForList: 'fdb2aafc-61a6-4cf0-aacd-2dcc93588e28',
                additionalInfo: '| mit Fries',
                course: 0,
                brangToTable: false,
                combinableWith: [
                    {
                        name: 'Fries',
                        price: '4.5',
                        active: true,
                        availableProducts: [
                            {
                                productId: 'Y0xzNAheX7OeQ703',
                                productName: 'Cheeseburger'
                            },
                            { productId: 'ZebYTu5kVZpk7jpO', productName: '
                        ],
                        combinationProduct: true,
                        _id: 'OoeIm5jDfzoA5ILm'
                    },
                    {
                        name: 'Potato Wedges',
                        price: '5',
                        active: true,
                        availableProducts: [
                            { productId: 'ZebYTu5kVZpk7jpO', productName: '
                            {
                                productId: 'Y0xzNAheX7OeQ703',
                                productName: 'Cheeseburger'
                            }
                        ],
                        combinationProduct: true,
                        _id: 'fbv0XiGlwfNmyQQn'
                    },
                    {
                        name: 'Chicken Wings',
```

```
                        price: '4.5',
                        active: true,
                        availableProducts: [
                            { productId: 'ZebYTu5kVZpk7jpO', productName: '
                            {
                                productId: 'Y0xzNAheX7OeQ703',
                                productName: 'Cheeseburger'
                            }
                        ],
                        combinationProduct: true,
                        _id: 'tB8sw3h9CA05xktr'
                    }
                ],
                combinedWith: [{ productName: 'Fries', price: '4.5' }]
            },
            {
                name: 'Margherita',
                price: '9.2',
                category: 'Pizza',
                tax: '19',
                kitchenRelevant: true,
                active: true,
                combinationProduct: false,
                infoText: '',
                stockChecking: false,
                stockAmount: 0,
                order: 0,
                customPrinterAddress: 'barPrinter',
                _id: '1ymAPOHWTFft5wFp',
                identifyForList: '3b38d0e3-3c4d-4396-86c7-9b6236570bb2',
                additionalInfo: '',
                course: 0,
                brangToTable: false,
                combinableWith: [],
                combinedWith: []
            }
        ]
    }),
})
    .then(response => response.json())
    .then(data => console.dir(data))
    .catch((error) => {
        console.error('Error:', error);
    });
```

Header

Feld	Beschreibung
Content-Type	application/json

Request Body

Feld	Typ	Beschreibung
table	**string**	Table number for the order. Please refere to /getTableNumber to get the amount of tables. If no table is chosen, this variable holds the value:'ohne Tisch' (no table)
articles	**array**	This array holds all articles that should be ordered for the specified table. Every article has the following properties:
articles._id	**string**	Unique identifier string assigned to each article (same as in a product)
articles.name	**string**	Name of the article (same as in a product)
articles.price	**string**	price of the article (same as in a product)
articles.category	**string**	Category of the article (same as in a product)
articles.tax	**string**	Value added tax rate required by law for the specified article (same as in a product)
articles.kitchenRelevant	**boolean**	Value to specify wheter the article should be sent to the kitchen printer (true) or the bar printer (false) and kitchen frontend View, when it is ordered. E.g. drinks are set to false, a Pizza or other dishes are set to true. (same as in a product)
articles.active	**boolean**	active boolean (same as in a product)
articles.combinationProduct	**boolean**	Specifies wheter the product is a combination product or not (can only be true at this point, because

		combination products cannot be ordered alone, only in combination with another product)
articles.infoText	**string**	String which contains additional information about the product. (same as in a product)
articles.order	**number**	not in use (same as in a product)
articles.stockChecking	**boolean**	Value to switch the stockChecking funktionality on and off. (same as in a product)
articles.stockAmount	**number**	If stockChecking is set to true, this value will determine how many products are left in the restaurants stock. (same as in a product)
articles.additionalInfo	**string**	This variable holds individual information about the ordered article. In case of the Cheeseburger, this field could for example contain 'medium'. (has to be added on top of the product values)
articles.course	**number**	The course variable can hold four different values 0-→Direkt (bring directly to the table), 1→Vorspeise (starter), 2→Hauptgang (main course), 3-->Nachspeise (dessert). This information will be rendered in the kitchen View at the frontend. (has to be added on top of the product values)
articles.brangToTable	**number**	This boolean holds the information wheter the article was already brought to the table and can be changed individually for every single article inside the order, please refere to /brangToTableUpdate.(has to be added on top of the product values)

articles.combinableWith	**array**	This array holds all combination products as objects, which are combinable with the current article. The structure of the combination product object is almost the same compared to an ordinary combination product:
articles.combinableWith.name	**string**	name of the combination product (see /getAllCombinationProducts)
articles.combinableWith.price	**string**	price of the combination product (see /getAllCombinationProducts)
articles.combinableWith.active	**array**	specifies the availability state of the combination product (see /getAllCombinationProducts)
articles.combinableWith.availableProducts	**array**	This array contains a list of products, that are combinable with the present combination product. In the example, the Cheese- and BBQ Burger are combinable with the Fries combination product. (see /getAllCombinationProducts)
articles.combinableWith.availableProducts.productId	**array**	ID of the product, which is combinable with the present combination product (see /getAllCombinationProducts)
articles.combinableWith.availableProducts.productName	**array**	Name of the product, which is combinable with the present combination product (see /getAllCombinationProducts)
articles.combinedWith	**array**	This array holds all combination products with their name and price as objects, which were actually combined with the ordered article, in this order. (In the example, the BBQ Burger was ordered with the Fries)
articles.combinedWith.productName	**array**	This value hold the name of the combination product, which should be added.
articles.combinedWith.price	**array**	This value holds the price of the combination product,

		which should be added.

200

Feld	Typ	Beschreibung
Order	**json**	If the request was successfull, the API will answer with the entire order object, which was added to the order file of the server. As already said, the new order object holds more properties, than just the added articles and the table number.

```
//the server answeres with the entire order object:
{
  table: '2',
  time: '28-11-2022 13:46:33',
  payment: 'BAR',
  orders: [
    {
      name: 'Cheeseburger',
      price: '24.5',
      category: 'Burger',
      tax: '19',
      kitchenRelevant: true,
      active: true,
      combinationProduct: false,
      infoText: '',
      stockChecking: false,
      stockAmount: 0,
      order: 0,
      customPrinterAddress: 'barPrinter',
      _id: 'Y0xzNAheX7OeQ703',
      identifyForList: '864e2fbc-515a-4833-ac76-c8c39e35e8cc',
      additionalInfo: '| mit Fries| mit Potato Wedges',
      course: 0,
      brangToTable: false,
      combinableWith: [
        {
          name: 'Fries',
          price: '4.5',
          active: true,
          availableProducts: [
            {
              productId: 'Y0xzNAheX7OeQ703',
              productName: 'Cheeseburger'
            },
            {
              productId: 'ZebYTu5kVZpk7jpO',
```

```
            productName: 'BBQ Burger'
          }
        ],
        combinationProduct: true,
        _id: 'OoeIm5jDfzoA5ILm'
      },
      {
        name: 'Potato Wedges',
        price: '5',
        active: true,
        availableProducts: [
          {
            productId: 'ZebYTu5kVZpk7jpO',
            productName: 'BBQ Burger'
          },
          {
            productId: 'Y0xzNAheX7OeQ703',
            productName: 'Cheeseburger'
          }
        ],
        combinationProduct: true,
        _id: 'fbv0XiGlwfNmyQQn'
      },
      {
        name: 'Chicken Wings',
        price: '4.5',
        active: true,
        availableProducts: [
          {
            productId: 'ZebYTu5kVZpk7jpO',
            productName: 'BBQ Burger'
          },
          {
            productId: 'Y0xzNAheX7OeQ703',
            productName: 'Cheeseburger'
          }
        ],
        combinationProduct: true,
        _id: 'tB8sw3h9CA05xktr'
      }
    ],
    combinedWith: [
      { productName: 'Fries', price: '4.5' },
      { productName: 'Potato Wedges', price: '5' }
    ],
    uniqueOrderArticleId: 'df5fcaac-b6bd-4b5e-a30e-23bc264ca958'
  },
```

```
    {
      name: 'BBQ Burger',
      price: '21',
      category: 'Burger',
      tax: '19',
      kitchenRelevant: true,
      active: true,
      combinationProduct: false,
      infoText: '',
      stockChecking: false,
      stockAmount: 0,
      order: 0,
      customPrinterAddress: 'barPrinter',
      _id: 'ZebYTu5kVZpk7jpO',
      identifyForList: 'fdb2aafc-61a6-4cf0-aacd-2dcc93588e28',
      additionalInfo: '| mit Fries',
      course: 0,
      brangToTable: false,
      combinableWith: [
        {
          name: 'Fries',
          price: '4.5',
          active: true,
          availableProducts: [
            {
              productId: 'Y0xzNAheX7OeQ703',
              productName: 'Cheeseburger'
            },
            {
              productId: 'ZebYTu5kVZpk7jpO',
              productName: 'BBQ Burger'
            }
          ],
          combinationProduct: true,
          _id: 'OoeIm5jDfzoA5ILm'
        },
        {
          name: 'Potato Wedges',
          price: '5',
          active: true,
          availableProducts: [
            {
              productId: 'ZebYTu5kVZpk7jpO',
              productName: 'BBQ Burger'
            },
            {
              productId: 'Y0xzNAheX7OeQ703',
```

```
            productName: 'Cheeseburger'
          }
        ],
        combinationProduct: true,
        _id: 'fbv0XiGlwfNmyQQn'
      },
      {
        name: 'Chicken Wings',
        price: '4.5',
        active: true,
        availableProducts: [
          {
            productId: 'ZebYTu5kVZpk7jpO',
            productName: 'BBQ Burger'
          },
          {
            productId: 'Y0xzNAheX7OeQ703',
            productName: 'Cheeseburger'
          }
        ],
        combinationProduct: true,
        _id: 'tB8sw3h9CA05xktr'
      }
    ],
    combinedWith: [ { productName: 'Fries', price: '4.5' } ],
    uniqueOrderArticleId: 'ddbb7cc8-1731-40ee-a32b-fa51e0469bb9'
  },
  {
    name: 'Margherita',
    price: '9.2',
    category: 'Pizza',
    tax: '19',
    kitchenRelevant: true,
    active: true,
    combinationProduct: false,
    infoText: '',
    stockChecking: false,
    stockAmount: 0,
    order: 0,
    customPrinterAddress: 'barPrinter',
    _id: '1ymAPOHWTFft5wFp',
    identifyForList: '3b38d0e3-3c4d-4396-86c7-9b6236570bb2',
    additionalInfo: '',
    course: 0,
    brangToTable: false,
    combinableWith: [],
    combinedWith: [],
```

```
      uniqueOrderArticleId: '2de13c86-60c7-4ddc-92f2-42ac32ebd988'
    }
  ],
  active: true,
  paidProducts: [],
  _id: 'DDr258KjStPi3ZcS'
}
```

Orders | Bring order to table (All users)

0.1.0

This is an API to get mark an article as "brought to table". In the backend, this API updates the "brangToTable" property of an selected article from an selected order and sets it to true. In the cash system frontend, articles then will be displayed with a green background, so that the staff knows that certain articles were already brought to the table. After the property was updated, the server will send the 'orderUpdate' message to all connected devices through the WebSocket channel.

POST

```
device_IP_Adress:3434/brangToTableUpdate
```

```
fetch('http://device_IP_Adress:3434/getAllActiveOrders/', {
    method: 'POST',
    headers: {
        'Content-Type': 'application/json',
    },
    body: JSON.stringify({
        orderId: "DDr258KjStPi3ZcS",
        productUniqueId: "df5fcaac-b6bd-4b5e-a30e-23bc264ca958"
    }),
})
    .then(response => response.json())
    .then(data => console.log(data))
    .catch((error) => {
        console.error('Error:', error);
    });
```

Header

Feld	Beschreibung
Content-Type	application/json

Request Body

Feld	Typ	Beschreibung
orderId	**string**	This string selects the order (by its ID) from which an article should be updated.
productUniqueId	**string**	This string selects the article, which should be marked as "brang to table".

200

Feld	Typ	Beschreibung
success	**none**	none

```
//the server answeres with all active orders as an array:
```

Orders | Delete order (All users)

0.1.0

This API is used to delete an order. After the order was removed successfully, the server will send the 'orderUpdate' message to all connected devices through the WebSocket channel.

POST

```
device_IP_Adress:3434/deleteOrder
```

```
fetch('http://device_IP_Adress:3434/deleteOrder/', {
    method: 'POST',
    headers: {
        'Content-Type': 'application/json',
    },
    body: JSON.stringify({
        orderId: "gitjLOsMHzKiid2T"
    }),
})
    .then(data => console.dir(data))
    .catch((error) => {
        console.error('Error:', error);
    });
```

Header

Feld	Beschreibung
Content-Type	application/json

Request Body

Feld	Typ	Beschreibung
orderId	**string**	This string selects the order (by its ID),which should be removed.

200

Feld	Typ	Beschreibung
success	**none**	none

Orders | Get all active orders (All users)

0.1.0

This is an API to get all the active orders. The API returns an array with all active orders as objects. An order objects contains an array called "orders", which contains all articles of each order, and additional information about the order.

GET

```
device_IP_Adress:3434/getAllActiveOrders
```

```
fetch('http://device_IP_Adress:3434/getAllActiveOrders/', {
    method: 'GET',
    headers: {
        'Content-Type': 'application/json',
    },
})
    .then(response => response.json())
    .then(data => console.log(data))
    .catch((error) => {
        console.error('Error:', error);
    });
```

Header

Feld	Beschreibung
Content-Type	application/json

Request Body

Feld	Typ	Beschreibung
table	**string**	This string holds the table number for which the order was made
time	**string**	This string holds the time, when the order was first ordered

payment	**string**	unused. default= "BAR"
orders	**array**	This array holds a list of all articles included in the order. To see an explanation for all the articles properties, please refere to /newOrder.
active	**boolean**	unused
paidProducts	**array**	This array holds a list of all articles, which have been paid. When adding a new order, each article gets assigned a new property called "uniqueOrderArticleId", which contains a string with a unique id. When using the payOrder API, the server will add the ID values of the articles, that have been payed.
_id	**string**	This string holds the orders unique id

200

Feld	Typ	Beschreibung
Order	**json**	If the request was successfull, the API will answer with the entire order object, which was added to the order file of the server. As already said, the new order object holds more properties, than just the added articles and the table number.

```
//the server answeres with all active orders as an array:
[
    {
        "table": "2",
        "time": "28-11-2022 13:46:33",
        "payment": "BAR",
        "orders": [
            {
                "name": "Cheeseburger",
                "price": "24.5",
                "category": "Burger",
                "tax": "19",
                "kitchenRelevant": true,
                "active": true,
                "combinationProduct": false,
                "infoText": "",
                "stockChecking": false,
                "stockAmount": 0,
                "order": 0,
                "customPrinterAddress": "barPrinter",
                "_id": "Y0xzNAheX7OeQ703",
                "identifyForList": "864e2fbc-515a-4833-ac76-c8c39e35e8cc",
                "additionalInfo": "| mit Fries| mit Potato Wedges",
                "course": 0,
                "brangToTable": false,
                "combinableWith": [
                    {
```

```
                "name": "Fries",
                "price": "4.5",
                "active": true,
                "availableProducts": [
                    {
                        "productId": "Y0xzNAheX7OeQ703",
                        "productName": "Cheeseburger"
                    },
                    {
                        "productId": "ZebYTu5kVZpk7jpO",
                        "productName": "BBQ Burger"
                    }
                ],
                "combinationProduct": true,
                "_id": "OoeIm5jDfzoA5ILm"
            },
            {
                "name": "Potato Wedges",
                "price": "5",
                "active": true,
                "availableProducts": [
                    {
                        "productId": "ZebYTu5kVZpk7jpO",
                        "productName": "BBQ Burger"
                    },
                    {
                        "productId": "Y0xzNAheX7OeQ703",
                        "productName": "Cheeseburger"
                    }
                ],
                "combinationProduct": true,
                "_id": "fbv0XiGlwfNmyQQn"
            },
            {
                "name": "Chicken Wings",
                "price": "4.5",
                "active": true,
                "availableProducts": [
                    {
                        "productId": "ZebYTu5kVZpk7jpO",
                        "productName": "BBQ Burger"
                    },
                    {
                        "productId": "Y0xzNAheX7OeQ703",
                        "productName": "Cheeseburger"
                    }
                ],
```

```
                        "combinationProduct": true,
                        "_id": "tB8sw3h9CA05xktr"
                    }
                ],
                "combinedWith": [
                    {
                        "productName": "Fries",
                        "price": "4.5"
                    },
                    {
                        "productName": "Potato Wedges",
                        "price": "5"
                    }
                ],
                "uniqueOrderArticleId": "df5fcaac-b6bd-4b5e-a30e-23bc264ca9
            },
            {
                "name": "BBQ Burger",
                "price": "21",
                "category": "Burger",
                "tax": "19",
                "kitchenRelevant": true,
                "active": true,
                "combinationProduct": false,
                "infoText": "",
                "stockChecking": false,
                "stockAmount": 0,
                "order": 0,
                "customPrinterAddress": "barPrinter",
                "_id": "ZebYTu5kVZpk7jpO",
                "identifyForList": "fdb2aafc-61a6-4cf0-aacd-2dcc93588e28",
                "additionalInfo": "| mit Fries",
                "course": 0,
                "brangToTable": false,
                "combinableWith": [
                    {
                        "name": "Fries",
                        "price": "4.5",
                        "active": true,
                        "availableProducts": [
                            {
                                "productId": "Y0xzNAheX7OeQ703",
                                "productName": "Cheeseburger"
                            },
                            {
                                "productId": "ZebYTu5kVZpk7jpO",
                                "productName": "BBQ Burger"
```

```
                    }
                ],
                "combinationProduct": true,
                "_id": "OoeIm5jDfzoA5ILm"
            },
            {
                "name": "Potato Wedges",
                "price": "5",
                "active": true,
                "availableProducts": [
                    {
                        "productId": "ZebYTu5kVZpk7jpO",
                        "productName": "BBQ Burger"
                    },
                    {
                        "productId": "Y0xzNAheX7OeQ703",
                        "productName": "Cheeseburger"
                    }
                ],
                "combinationProduct": true,
                "_id": "fbv0XiGlwfNmyQQn"
            },
            {
                "name": "Chicken Wings",
                "price": "4.5",
                "active": true,
                "availableProducts": [
                    {
                        "productId": "ZebYTu5kVZpk7jpO",
                        "productName": "BBQ Burger"
                    },
                    {
                        "productId": "Y0xzNAheX7OeQ703",
                        "productName": "Cheeseburger"
                    }
                ],
                "combinationProduct": true,
                "_id": "tB8sw3h9CA05xktr"
            }
        ],
        "combinedWith": [
            {
                "productName": "Fries",
                "price": "4.5"
            }
        ],
        "uniqueOrderArticleId": "ddbb7cc8-1731-40ee-a32b-fa51e0469b
```

```
            },
            {
                "name": "Margherita",
                "price": "9.2",
                "category": "Pizza",
                "tax": "19",
                "kitchenRelevant": true,
                "active": true,
                "combinationProduct": false,
                "infoText": "",
                "stockChecking": false,
                "stockAmount": 0,
                "order": 0,
                "customPrinterAddress": "barPrinter",
                "_id": "1ymAPOHWTFft5wFp",
                "identifyForList": "3b38d0e3-3c4d-4396-86c7-9b6236570bb2",
                "additionalInfo": "",
                "course": 0,
                "brangToTable": false,
                "combinableWith": [],
                "combinedWith": [],
                "uniqueOrderArticleId": "2de13c86-60c7-4ddc-92f2-42ac32ebd9
            }
        ],
        "active": true,
        "paidProducts": [
            "ddbb7cc8-1731-40ee-a32b-fa51e0469bb9",
            "2de13c86-60c7-4ddc-92f2-42ac32ebd988"
        ],
        "_id": "DDr258KjStPi3ZcS"
    },
    {
        "table": "ohne Tisch",
        "time": "28-11-2022 18:56:2",
        "payment": "BAR",
        "orders": [
            {
                "name": "Margherita",
                "price": "9.2",
                "category": "Pizza",
                "tax": "19",
                "kitchenRelevant": true,
                "active": true,
                "combinationProduct": false,
                "infoText": "",
                "stockChecking": false,
                "stockAmount": 0,
```

```
                "order": 0,
                "customPrinterAddress": "barPrinter",
                "_id": "1ymAPOHWTFft5wFp",
                "identifyForList": "6b245635-1c56-48fb-be24-fac5e75e1de7",
                "additionalInfo": "",
                "course": 0,
                "brangToTable": false,
                "combinableWith": [],
                "combinedWith": [],
                "uniqueOrderArticleId": "6aaa1aae-6b1c-4a01-a6a5-0234350197
            }
        ],
        "active": true,
        "paidProducts": [],
        "_id": "xiyq2dzF2zPTN8lW"
    }
]
```


Orders | Pay order (All users)

0.1.0

This is an API to pay orders. Since the name is a bit misleading, it has to be said, that it is also possible to only pay different parts of orders, instead of the whole order. After sending a payOrder request to the cash, the cash will start different processes in the background. Although the user doesn't have to worry about them, to provide a better understanding, they will be stated here. First the cash starts a mandatory TSS transaction, which is required by law, to save the payment on another service providing platform. Furthermore the cash will print out a receipt with the specified printer. When these steps are done, the cash will update the order file of the cash system and add the paid articles to the paid products array of the order.

POST

```
device_IP_Adress:3434/payOrder
```

```
fetch('http://device_IP_Adress:3434/payOrder/', {
    method: 'POST',
    headers: {
        'Content-Type': 'application/json',
    },
    body: JSON.stringify({
        table: "2",
        responsibleEmployee: "Frederik",
        paymentMethod: "BAR",
        Qrcode: false,
        startTime: "29-11-2022 11:30:13",
        timestamp: Date.now(),
```

```
        tipValue: 5,
        total: 54.7,
        orderId: "37J6f43wKcfd3qqw",
        payedArticels: [
            {
                name: 'Cheeseburger',
                price: '24.5',
                category: 'Burger',
                tax: '19',
                kitchenRelevant: true,
                active: true,
                combinationProduct: false,
                infoText: '',
                stockChecking: false,
                stockAmount: 0,
                order: 0,
                customPrinterAddress: 'barPrinter',
                _id: 'Y0xzNAheX7OeQ703',
                identifyForList: '91e2e30f-cce1-4b3e-bf7e-e5acda30a3bb',
                additionalInfo: '| mit Fries| mit Potato Wedges',
                course: 0,
                brangToTable: false,
                combinableWith: [
                    {
                        name: 'Fries',
                        price: '4.5',
                        active: true,
                        availableProducts: [
                            {
                                productId: 'Y0xzNAheX7OeQ703',
                                productName: 'Cheeseburger'
                            },
                            { productId: 'ZebYTu5kVZpk7jpO', productName: '
                        ],
                        combinationProduct: true,
                        _id: 'OoeIm5jDfzoA5ILm'
                    },
                    {
                        name: 'Potato Wedges',
                        price: '5',
                        active: true,
                        availableProducts: [
                            { productId: 'ZebYTu5kVZpk7jpO', productName: '
                            {
                                productId: 'Y0xzNAheX7OeQ703',
                                productName: 'Cheeseburger'
                            }
```

```
                    ],
                    combinationProduct: true,
                    _id: 'fbv0XiGlwfNmyQQn'
                },
                {
                    name: 'Chicken Wings',
                    price: '4.5',
                    active: true,
                    availableProducts: [
                        { productId: 'ZebYTu5kVZpk7jpO', productName: '
                        {
                            productId: 'Y0xzNAheX7OeQ703',
                            productName: 'Cheeseburger'
                        }
                    ],
                    combinationProduct: true,
                    _id: 'tB8sw3h9CA05xktr'
                }
            ],
            combinedWith: [
                { productName: 'Fries', price: '4.5' },
                { productName: 'Potato Wedges', price: '5' }
            ],
            uniqueOrderArticleId: '2ed997ed-c78f-47ff-af91-448c6fa680ec
        },
        {
            name: 'BBQ Burger',
            price: '21',
            category: 'Burger',
            tax: '19',
            kitchenRelevant: true,
            active: true,
            combinationProduct: false,
            infoText: '',
            stockChecking: false,
            stockAmount: 0,
            order: 0,
            customPrinterAddress: 'barPrinter',
            _id: 'ZebYTu5kVZpk7jpO',
            identifyForList: 'effc6b88-b70d-49dc-9ebd-1832aa7a6a59',
            additionalInfo: '| mit Fries',
            course: 0,
            brangToTable: false,
            combinableWith: [
                {
                    name: 'Fries',
                    price: '4.5',
```

```
                    active: true,
                    availableProducts: [
                        {
                            productId: 'Y0xzNAheX7OeQ703',
                            productName: 'Cheeseburger'
                        },
                        { productId: 'ZebYTu5kVZpk7jpO', productName: '
                    ],
                    combinationProduct: true,
                    _id: 'OoeIm5jDfzoA5ILm'
                },
                {
                    name: 'Potato Wedges',
                    price: '5',
                    active: true,
                    availableProducts: [
                        { productId: 'ZebYTu5kVZpk7jpO', productName: '
                        {
                            productId: 'Y0xzNAheX7OeQ703',
                            productName: 'Cheeseburger'
                        }
                    ],
                    combinationProduct: true,
                    _id: 'fbv0XiGlwfNmyQQn'
                },
                {
                    name: 'Chicken Wings',
                    price: '4.5',
                    active: true,
                    availableProducts: [
                        { productId: 'ZebYTu5kVZpk7jpO', productName: '
                        {
                            productId: 'Y0xzNAheX7OeQ703',
                            productName: 'Cheeseburger'
                        }
                    ],
                    combinationProduct: true,
                    _id: 'tB8sw3h9CA05xktr'
                }
            ],
            combinedWith: [{ productName: 'Fries', price: '4.5' }],
            uniqueOrderArticleId: 'c8ae8814-c25d-4651-a866-fc64795a153f
        },
        {
            name: 'Margherita',
            price: '9.2',
            category: 'Pizza',
```

```
                tax: '19',
                kitchenRelevant: true,
                active: true,
                combinationProduct: false,
                infoText: '',
                stockChecking: false,
                stockAmount: 0,
                order: 0,
                customPrinterAddress: 'barPrinter',
                _id: '1ymAPOHWTFft5wFp',
                identifyForList: 'c39c08e5-b723-4ad5-9f89-25aec9eaed12',
                additionalInfo: '',
                course: 0,
                brangToTable: false,
                combinableWith: [],
                combinedWith: [],
                uniqueOrderArticleId: 'ea3a22db-516f-4aa6-b8fa-ea0d1fa72aff
            }
        ]

    },
})
    .then(response => response.json())
    .then(data => console.dir(data))
    .catch((error) => {
        console.error('Error:', error);
    });
```

Header

Feld	Beschreibung
Content-Type	application/json

Request Body

Feld	Typ	Beschreibung
orderId	**string**	This string holds the ID of the order from which articles should be payed.
payedArticels	**array**	This array holds a list of the articles that should be payed. These articles have the exact same structure as the articles from /getAllActiveOrders
paymentMethod	**string**	This string holds the payment method of the paying process, which can be "BAR" or "KARTE"
tipValue	**string**	This string holds the amount of the tip that was given (if none this holds "0")

responsibleEmployee	**string**	This string holds the name of the employee that made the payment
timestamp	**number**	This string holds holds the current timestamp (when the pay order request is send by a user)
table	**string**	This string holds the table number ("ohne Tisch" in case of no specified table)
startTime	**string**	This string holds start time of the order (when the order was ordered by an employee)
total	**number**	This number holds the total price of the order.
Qrcode	**boolean**	unused (should be set to false, otherwise the printer won't print a receipt)

200

Feld	Typ	Beschreibung
Order	**json**	If the request was successfull, the API will answer with the entire order object, which was added to the order file of the server. As already said, the new order object holds more properties, than just the added articles and the table number.

```
//the server answeres with the entire order object:
```

Orders | Remove article from order (All users)

0.1.0

This API is used to remove selected articles from an order. After the article was removed successfully , the server will send the 'orderUpdate' message to all connected devices through the WebSocket channel.

POST

```
device_IP_Adress:3434/removeFromOrder
```

```
fetch('http://device_IP_Adress:3434/removeFromOrder/', {
    method: 'POST',
    headers: {
        'Content-Type': 'application/json',
    },
    body: JSON.stringify({
        orderId: "DDr258KjStPi3ZcS",
        productUniqueId: "df5fcaac-b6bd-4b5e-a30e-23bc264ca958"
    }),
})
    .then(response => response.json())
    .then(data => console.log(data))
```

```
    .catch((error) => {
        console.error('Error:', error);
    });
```

Header

Feld	Beschreibung
Content-Type	application/json

Request Body

Feld	Typ	Beschreibung
orderId	**string**	This string selects the order (by its ID) from which an article should be removed.
productUniqueId	**string**	This string selects the article, which should be removed.

200

Feld	Typ	Beschreibung
success	**none**	none

Products

Products | Add new combination product (Admin)

0.1.0

API to add a new combination product to the cash system. After the combination product was added, the server will send the 'productUpdate' message to all connected devices through the WebSocket channel.

POST

```
device_IP_Adress:3434/newCombinationProduct
```

```
fetch('http://device_IP_Adress:3434/newCombinationProduct/', {
    method: 'POST',
    headers: {
        'Content-Type': 'application/json',
    },
    body: JSON.stringify({
        name: 'Chicken Wings',
        price: '4.5',
        "availableProducts": [
```

```
            {
                "productId": "c3SWGAN6gCxVMUoJ",
                "productName": "Cheeseburger"
            },
            {
                "productId": "WkMGc1A2pwhMnc8V",
                "productName": "BBQ Burger"
            }
        ],
    }),
})
    .then(response => response.json())
    .then(data => console.log(data))
    .catch((error) => {
        console.error('Error:', error);
    });
```

Header

Feld	Beschreibung
Content-Type	application/json

Request Body

Feld	Typ	Beschreibung
name	**string**	name of product to add
price	**string**	price of product to add
availableProducts	**array**	This array contains a list of products, that are combinable with the combination product which should be added. In the example, the Cheese- and BBQ Burger are combinable with the potato wedges combination product.
availableProducts.productId	**string**	ID of the product, which is combinable with the combination product
availableProducts.productName	**string**	Name of the product, which is combinable with the combination product

200

Feld	Typ	Beschreibung
ID	**json**	The API will answer with the ID which is added to the new product by the server.

Products | Add new product (Admin)

0.1.0

API to add a new ordinary product to the cash system. After the product was added, the server will send the 'productUpdate' message to all connected devices through the WebSocket channel. The server will also add some variables to the products, please refere to the documentation of /getAllProducts, to see all the values of a product.

POST

```
device_IP_Adress:3434/newProduct
```

```
fetch('http://device_IP_Adress:3434/newProduct/', {
    method: 'POST',
    headers: {
        'Content-Type': 'application/json',
    },
    body: JSON.stringify({
        name: 'Speciale',
        price: '11',
        category: 'Pizza',
        tax: '19',
        kitchenRelevant: true,
        infoText: '',
        stockChecking: false,
        stockAmount: 0
    }),
})
    .then(response => response.json())
    .then(data => console.log(data))
    .catch((error) => {
        console.error('Error:', error);
    });
```

Header

Feld	Beschreibung
Content-Type	application/json

Request Body

Feld	Typ	Beschreibung
name	**string**	name of product to add
price	**string**	price of product to add

category	**string**	category of product to add
tax	**string**	tax rate of product to add
kitchenRelevant	**boolean**	kitchenRelevant boolean of product to add
infoText	**string**	infoText of product to add
stockChecking	**boolean**	stockChecking boolean of product to add
stockAmount	**number**	stockAmount of product to add

200

Feld	Typ	Beschreibung
ID	**json**	The API will answer with the ID which is added to the new product by the server.

Products | Delete a product (Admin)

0.1.0

API to delte a product from the devices product file. This works for every product, no matter if it is a combination or an ordinary product. After a successfull update, the server will send a 'productUpdate' WebSocket message.

POST

```
device_IP_Adress:3434/deleteProduct
```

```
fetch('http://device_IP_Adress:3434/deleteProduct/', {
  method: 'POST',
  headers: {
     'Content-Type': 'application/json',
  },
    body: JSON.stringify({
        _id: 'eb1wAq8GBZ2pIkob',
    }),
  .then(response => console.log(response.status))
    .catch((error) => {
       console.error('Error:', error);
    });
```

Header

Feld	Beschreibung
Content-Type	application/json

Request Body

Feld	Typ	Beschreibung
_id	**string**	Unique identifier of chosen product to delete

200

Feld	Typ	Beschreibung
Statuscode	**none**	The API will return status code 200 if the request was successfull

Products | Get all combination products (All users)

0.1.0

API to get all the combination products the restaurant has to offer. The API returns an array with combination product Objects, each containing the mandatory properties of a combination product e.g. name, price etc.

GET

```
device_IP_Adress:3434/getAllCombinationProducts
```

```
  fetch('http://device_IP_Adress:3434/getAllCombinationProducts/', {
  method: 'GET',
  headers: {
    'Content-Type': 'application/json',
  }
})
  .then(response => response.json())
  .then(data => {
    if (data != null) {
        console.log(data)
    }
  })
  .catch((error) => {
    console.error('Error:', error);
  });
```

Header

Feld	Beschreibung
Content-Type	application/json

200

Feld	Typ	Beschreibung
products	**array**	Returned Array with all products as distinc objects
_id	**string**	Unique identifier string assigned to each product
name	**string**	Name of the product
price	**string**	price of the product
active	**string**	If a product is not available anymore (for whatever reason), this value can be set to false, hence the product won't be accessable in the order controller view. (the staff won't be able to order this product)
combinationProduct	**boolean**	Specifies whether the product is a combination product or not
availableProducts	**array**	This array contains a list of products, that are combinable with the present combination product. In the example, the Cheese- and BBQ Burger are combinable with the potato wedges combination product.
productId	**string**	ID of the product, which is combinable with the present combination product
productName	**string**	Name of the product, which is combinable with the present combination product

```
[
    {
        "name": "Fries",
        "price": "4.5",
        "active": true,
        "availableProducts": [
            {
                "productId": "c3SWGAN6gCxVMUoJ",
                "productName": "Cheeseburger"
            },
            {
                "productId": "WkMGc1A2pwhMnc8V",
                "productName": "BBQ Burger"
            }
        ],
        "combinationProduct": true,
        "_id": "0b7Y6etiNnHyHZTc"
    },
    {
        "name": "Potato Wedges",
        "price": "5",
        "active": true,
        "availableProducts": [
            {
                "productId": "c3SWGAN6gCxVMUoJ",
                "productName": "Cheeseburger"
            },
```

```
            {
                "productId": "WkMGc1A2pwhMnc8V",
                "productName": "BBQ Burger"
            }
        ],
        "combinationProduct": true,
        "_id": "kThC1Fqr3Qedc4tQ"
    }
]
```

Products | Get all Products (All users)

0.1.0

API to get all the products the restaurant has to offer, even the unavailable and sold out ones. The API returns an array with product Objects, each containing the mandatory properties of a product e.g. name, price etc. Products which can be added on top of other products (combination products) are excluded in the returned array, but can be fetched with /getAllCombinationProducts. The properties of each product can be set by an admin user using the /updateProduct request.

GET

```
device_IP_Adress:3434/getAllProducts
```

```
  fetch('http://device_IP_Adress:3434/getAllProducts/', {
  method: 'GET',
  headers: {
    'Content-Type': 'application/json',
  }
})
  .then(response => response.json())
  .then(data => {
    if (data != null) {
        console.log(data)
    }
  })
  .catch((error) => {
    console.error('Error:', error);
  });
```

Header

Feld	Beschreibung
Content-Type	application/json

200

Feld	Typ	Beschreibung
products	**array**	Returned Array with all products as distinc objects
_id	**string**	Unique identifier string assigned to each product
name	**string**	Name of the product
price	**string**	price of the product
category	**string**	Category of the product
tax	**string**	Value added tax rate required by law for the specified product
kitchenRelevant	**boolean**	Value to specify wheter the product should be sent to the kitchen Printer and kitchen frontend View, when it is ordered. E.g. drinks are set to false, a Pizza or other dishes are set to true.
active	**string**	If a product is not available anymore (for whatever reason), this value can be set to false, hence the product won't be accessable in the order controller view. (the staff won't be able to order this product)
combinationProduct	**boolean**	Specifies wheter the product is a combination product or not
infoText	**string**	String which contains additional informatioN about the product. Regarding a coctail, this empty string can be substituted with the description of the coctail or how to mix it. This information string will be rendered in the kitchenView for example, when the question mark button is clicked.
order	**number**	not in use
stockChecking	**boolean**	Value to switch the stockChecking funktionality on and off. If this value is set to true, the active variable won't be used anymore and the availabilty of a product will be handled be the stockAmount variable
stockAmount	**number**	If stockChecking is set to true, this value will determine how many products are left in the restaurants stock. When a product with stockChecking ture is ordered, this variable will decrease by one. To make sure, that every connected device will get noticed by this decrease, use the /updatestockAmount API to update the stockAmount variable or the /incrementStockAmount API in case you want to inscrease it.


```
[
  {
    name: 'Margherita',
    price: '9.20',
    category: 'Pizza',
    tax: '19',
    kitchenRelevant: true,
    active: true,
    combinationProduct: false,
    infoText: '',
    order: 0,
```

```
    _id: '1wVOIebmvD8Ux8Ai',
    stockChecking: false,
    stockAmount: 0
  },
  {
    name: 'Spaghetti Bolognese',
    price: '8',
    category: 'Pasta',
    tax: '19',
    kitchenRelevant: true,
    active: true,
    combinationProduct: false,
    infoText: '',
    order: 0,
    _id: 'b3ABoglryZ9EK3Vp',
    stockChecking: false,
    stockAmount: 0
  },
  {
    name: 'Funghi',
    price: '10',
    category: 'Pizza',
    tax: '19',
    kitchenRelevant: true,
    active: true,
    combinationProduct: false,
    infoText: '',
    order: 0,
    _id: 'eb1wAq8GBZ2pIkob',
    stockChecking: false,
    stockAmount: 0
  }
]
```

Products | Increment stock amount(All users)

0.1.0

API to increment the stockamount property of a specified product by one. After the stockAmount was incremented, the server will send the 'productUpdate' Message to all connected devices through the WebSocket channel.

POST

```
device_IP_Adress:3434/incrementStockAmount
```

```
fetch('http://device_IP_Adress:3434/updateProduct/', {
    method: 'POST',
    headers: {
    'Content-Type': 'application/json',
    }
    body: JSON.stringify({
        _id: 'eb1wAq8GBZ2pIkob',
    }),
  .then(response => console.log(response.status))
    .catch((error) => {
    console.error('Error:', error);
    });
```

Header

Feld	Beschreibung
Content-Type	application/json

Request Body

Feld	Typ	Beschreibung
_id	**string**	Unique identifier of chosen product to increment the stockAmount variable by one

200

Feld	Typ	Beschreibung
Statuscode	**none**	The API will return status code 200 if the request was successfull

Products | Update combination product (Admin)

0.1.0

API to update the values of a specified combination product. After the combination product was updated, the server will send the 'productUpdate' message to all connected devices through the WebSocket channel.

POST

```
device_IP_Adress:3434/updatingComboArticle
```

```
// in this example the potato wedges should only be compatible with the che
fetch('http://device_IP_Adress:3434/updatingComboArticle/', {
    method: 'POST',
```

```
    headers: {
    'Content-Type': 'application/json',
    },
    body: JSON.stringify({
        _id: 'kThC1Fqr3Qedc4tQ',
        name: 'Potato Wedges',
        price: '5',
        active: true,
        availableProducts:[
            {
                "productId": "c3SWGAN6gCxVMUoJ",
                "productName": "Cheeseburger"
            },
        ],
    }),
})
   .then(response => console.log(response.status))
    .catch((error) => {
    console.error('Error:', error);
    });
```

Header

Feld	Beschreibung
Content-Type	application/json

Request Body

Feld	Typ	Beschreibung
_id	**string**	Unique identifier of the combination product to update
name	**string**	updated name of the combination product to update
price	**string**	updated price of the combination product to update
active	**boolean**	updated active boolean of the combination product to update
availableProducts	**array**	updated array of products that are combinable with this combination product

200

Feld	Typ	Beschreibung
Statuscode	**none**	The API will return status code 200 if the request was successfull

Products | Update stock amount(All users)

0.1.0

API to update the stockamount property of a specified product. After the stockAmount was updated, the server will send the 'productUpdate' Message to all connected devices through the WebSocket channel.

POST

```
device_IP_Adress:3434/updatestockAmount
```

```
fetch('http://device_IP_Adress:3434/updateProduct/', {
    method: 'POST',
    headers: {
    'Content-Type': 'application/json',
    }
    body: JSON.stringify({
        _id: 'eb1wAq8GBZ2pIkob',
        stockAmount: 12
    }),
  .then(response => console.log(response.status))
    .catch((error) => {
    console.error('Error:', error);
    });
```

Header

Feld	Beschreibung
Content-Type	application/json

Request Body

Feld	Typ	Beschreibung
_id	**string**	Unique identifier of chosen product to update
stockAmount	**number**	new stockAmount value of the chosen product

200

Feld	Typ	Beschreibung
Statuscode	**none**	The API will return status code 200 if the request was successfull

Connecting from the Internet

0.1.0

All the HTTP Requests are available from the Internet if the cash system is online and running. Just the URL is a bit different

POST

```
https://router.degaso.de:8080/route/[CUSTOMER_KEY]/[REQUEST]/
```