API – Documentation

**Authentication**

**Endpoint overview:**

POST /auth/users

GET /auth/user

GET /auth/login

POST /auth/local

DELETE /auth/local

POST /auth/google

**The authentication object:**

Attributes:

* username: username
* groups: what kind of user it is (admin, user)

The authentication object can look like this:

{  
 "username": "OlaNordmann",  
 "groups": “user”  
}

**Create a user:** POST /auth/users

**Retrieve a user:** GET /auth/user

**Log in:** GET /auth/login

**Checkout**

**Endpoint overview:**

POST /api/vi/checkout

**The checkout object:**

Attributes:

* products: product object that has two child attributes:
  + product\_id: unique id
  + amount: the amount of this product added

The checkout object can look like this:

{  
 "products": [  
 {  
 "product\_id": 343423243535,  
 "amount": 3  
 }  
 ]  
}

**Create a checkout:** POST /api/vi/checkout

Creates a new product object

Returns:

Object with three attributes; a message, session expire date and checkout ID, on success. Otherwise, this call returns an error.

**Discount**

**Endpoint overview:**

POST /api/v1/discounts

GET /api/v1/discounts

GET /api/v1/discount/:discount\_code

**The discount object:**

Attributes:

* code: discount code
* percent: how many percent discount the code gives
* user: Specify which users get this as a valid code (optional)
* products: Specify which products get this as a valid code (optional)

The object can look like this:

{  
 "code": "Alfred",  
 "percent": 100,  
 "user": null,  
 "products": null  
}

**Create a discount:** POST /api/v1/discounts

Creates a new discount object (see above)

Policies:

* Admin

Returns:

Object with a message on success. Otherwise, this call returns an error.

**Retrieve all discounts:** GET /api/v1/discounts

Policies:

* Admin

Returns:

A list of valid discount codes

**Retrieve a discount:** GET /api/v1/discount/:discount\_code

Policies:

* User

Parameters:

* discount\_code: valid discount code

Returns:

The same object as above

**Orders**

**Endpoint overview:**

GET /api/v1/order/:order\_id

GET /api/v1/orders

DELETE /api/v1/order/:order\_id

POST /api/v1/orders

**The order object:**

Attributes:

* checkout\_session: session ID
* payment\_token: payment token
* cupon: active valid cupon code (optional)
* note: information about order (optional)
* address:
  + country:
  + province: province
  + city: city
  + zip: zip-number
  + street: name of street and number

The discount object can look like this:

{  
 "checkout\_session": 12345678,  
 "payment\_token": erferf788f7erfwe,  
 "cupon": null,  
 "note": null,  
 "address": {  
 "country": Norway,  
 "province": Oslo,  
 "city": Oslo,  
 "zip": 1234,  
 "street": Osloveien 12,  
 }  
}

**Retrieve an order:** GET /api/v1/order/:order\_id

Retrieves the details of an existing order. Supply the unique order ID from either an order creation request or the order list.

Policies:

* Owner
* Admin

Parameters:

* order\_id

Returns:

An order object if a valid identifier was provided.

**Retrieve all orders:** GET /api/v1/orders

Returns a list of your orders.

Policies:

* Owner

Query:

* user\_id (optional)

Returns:

List of your orders

**Delete an order:** DELETE /api/v1/order/:order\_id

Policies:

* Admin
* Owner

Parameter:

* order id

Returns:

A message object on success. Otherwise, this call returns an error.

**Create an order:** POST /api/v1/orders

Creates a new order object

Returns:

An object with a message and order\_id on success. Otherwise, this call returns an error.

**Products**

**Endpoint overview:**

POST /api/v1/products

GET /api/v1/product/:product\_id

GET /api/v1/products

PUT /api/v1/product/:product\_id

DELETE /api/v1/product/:product\_id

**The product object:**

Attributes:

* name: name of product
* price: price of product
* stock: how many available
* keywords: List of strings describing product
* short\_desc: Short description of product
* long\_desc: Long description of product

The product object will look like this:

{  
 "name": "go pro",  
 "price": 2200,  
 "stock": 23,  
 "keywords": “camera”,  
 "short\_desc": "Camera for outdoor activities",  
 "long\_desc": “Long lasting camera that is perfect for all kinds of activities. It can be submerged in water.”  
}

**Create a product:** POST /api/v1/products

Creates a new product object (see above)

Policies: Admin

Returns an object with a message and product\_id on success. Otherwise, this call returns an error.

**Retrieve a product:** GET /api/v1/product/:product\_id

Retrieves the details of an existing product. Supply the unique product ID from either a product creation request or the product list

Parameters:

* product\_id: unique product identifier

Returns:

Returns a product object if a valid identifier was provided

**Retrieve all products:** GET /api/v1/products

Returns a list of products

Query (all optional):

* index: specify starting index for products
* count: specify how many products from index you want
* keyword: only list products with containing set keyword
* search: only list products that match search word

The object will look like this:

{  
 "index": 0  
 "count": 20  
 "keyword": “phone”,  
 "search": “iPhone”  
}

Return:

List for products matching query or all products if no queries are specified.

**Update a product**

Updates the specific product by setting the values of the object attributes passed. Any attribute not provided will be left unchanged.

Policies: Admin

Attributes that can be specified/changed (all optional):

* name: name of product
* price: price of product
* keywords: list of strings describing product
* short\_desc: short description of product
* long\_desc: long description of product

Body can look like this:

{  
 "name": "go pro",  
 "price": 1500,  
 "keywords": “camera”,  
 "short\_desc": "Camera for outdoor and indoor activities",  
 "long\_desc": “Long lasting camera that is perfect for all kinds of activities. It can be submerged in water and sand. Very rugged”  
}

Return:

Object with message if product was successfully updated

**Delete a product:** DELETE /api/v1/product/:product\_id

Delete a product with given product ID

Policies: Admin

Parameter:

* product\_id: product identifier

Return:

Object with message if product was successfully deleted

**Users**

**Endpoint overview:**

GET /api/v1/users

GET /api/v1/user/:user\_id

PUT /api/v1/user/:user\_id

DELETE /api/v1/user/:user\_id

**The user object:**

Attributes:

* username: unique name of user
* groups: what kind of user it is (admin, user).
* email: users email address.

The user object will look like this:

{  
 "username": "OlaNordmann",  
 "groups": “admin”  
 "email": "ola@nordmann.no"  
}

**Retrieve all users:** GET /api/v1/users

No parameters

Return:

List off all users

**Retrieve a user:** GET /api/v1/user/:user\_id

Retrieves the details of an existing user. Supply the unique user ID from either a user creation request or the user list

Parameter:

* user\_id: unique user ID

Return:

User object if a valid identifier was provided.

**Update a user:** PUT /api/v1/user/:user\_id

Updates the specific user by setting the values of the attributes passed. Any attribute not provided will be left unchanged.

Policies:

* owner
* admin

Parameter:

* user\_id: unique suer identifier

Attributes that can be specified/changed (all optional):

* username: username of user
* group: what kind of user it is
* email: users email

Body can look like this:

{  
 "username": null,  
 "group": null,  
 "email": “ola@gmail.com”,  
}

Return:

Object with message if product was successfully updated

**Delete a product:** DELETE /api/v1/user/:user\_id

Delete a user with given user ID

Policies:

* Admin
* Owner

Parameter:

* user\_id: user identifier

Return:

Object with message if user was successfully deleted

**Validator**

**Endpoint overview:**

POST /api/v1/validate/username

POST /api/v1/validate/password

POST /api/v1/validate/group

POST /api/v1/validate/email

POST /api/v1/validate/name

POST /api/v1/validate/price

POST /api/v1/validate/keyword

POST /api/v1/validate/shrot\_desc

POST /api/v1/validate/long\_desc

POST /api/v1/validate/count

POST /api/v1/validate/search\_query

POST /api/v1/validate/note

POST /api/v1/validate/country

POST /api/v1/validate/province

POST /api/v1/validate/city

POST /api/v1/validate/zip

POST /api/v1/validate/street\_name

POST /api/v1/validate/number

POST /api/v1/validate/product\_id

POST /api/v1/validate/order\_id

POST /api/v1/validate/checkout\_id

POST /api/v1/validate/payment\_token

POST /api/v1/validate/discount\_code

POST /api/v1/validate/percent

POST /api/v1/validate/milliseconds

POST /api/v1/validate/amount

All endpoints require one attribute containing value that needs to be validated

Body will look like this:

{  
 "endpoint": "value"  
}

Let’s take email as an example:

{

"email": "ola@nordmann.no"

}