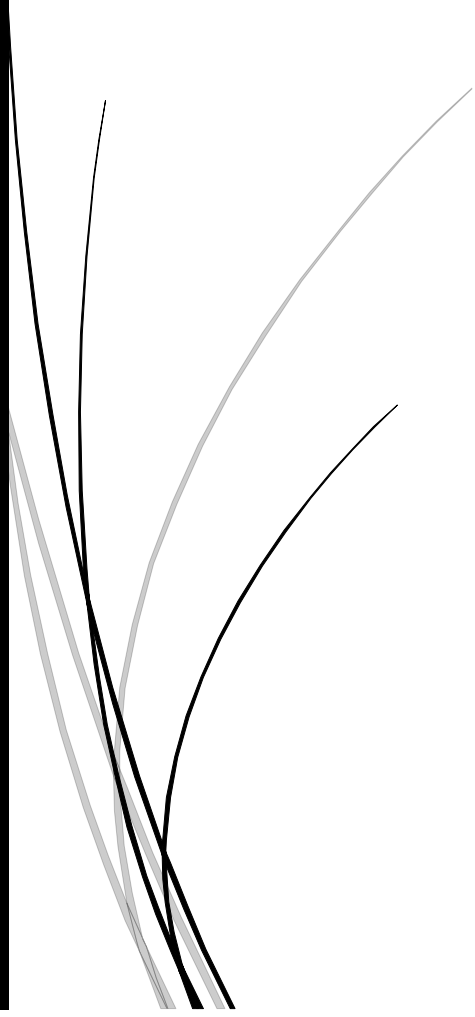
10/5/2020

FINA WEB API 3.1

დოკუმენტაცია



შპს FINA

სარჩევი

FINA Web API აღწერა

მისამართი (ოპერაციული მხარე): http://[IP:PORT]/api/operation

(რეპორტინგი): http://[IP:PORT]/api/reporting

(ავტორიზაცია): http://[IP:PORT]/api/authentication

Request Headers:

Content-Type:	application/json
Accept:	application/json
Authorization:	Bearer <access token>

დაბრუნებული სტატუს კოდები:

200 OK
400 Bad Request
500 Internal Server Error
401 Unauthorized

ცვლილებები:

ვერსია	მეთოდი	აღწერა
3.1.20201005	saveDocBonusOperation getBonusCardRestByCode getLoyaltyCardsByHolder getBonusCoeff	**დაემატა ახალი მეთოდი**
3.1.20200921	getPackedProducts	**დაემატა ახალი მეთოდი**
	authenticate	**ტოკენის მოქმედების ვადა შემცირდა 36 სთ-მდე.**
3.1.20200828	getProductsReserve	**გაუქმდა / ჩანაცვლდა მეთოდით -** getProductsRestArray
	getDocProductOutSingle getStoreGroups getProductsRestArray getProductsBarcodeArray	**დაემატა ახალი მეთოდი.**
3.1.20200808	getCharacteristicValuesArray	**დაემატა ახალი მეთოდი**.
3.1.20200617	getProductPricesAdvance	**დაემატა ახალი მეთოდი**.
3.1.20200605	getDocCustomerReturn getDocProductOut getDocProductMove getDocInventoryMove getDocInventoryOut getDocCustomerInventoryReturn getRealizesJournal getMovesJournal getCustomersReturnJournal getEntriesJournal	**დაემატა ველი:** waybill_num; **ერთმანეთისგან გაიმიჯნა ველები:** doc_num და waybill_num.
	getProducts	**დაემატა ველები:** web_group_id, order_id.
	getCharacteristics, getCharacteristicValues, getWebProductGroups, getProductsArray, getProductsImageArray, getProductsAfter	**დაემატა ახალი მეთოდები**.
	getProductGroups, getProvidedServiceGroups, getReceivedServiceGroups, getInventoryGroups,	**დაემატა ველი:** order_id.
3.1.20200520	getProductPrices	**დაემატა ველები:** discount_price, discount_start, discount_end.

ავტორიზაცია:

authenticate - ავტორიზაცია

- **ფუნქცია:** authenticate

  აღწერა: ავტორიზაცია (ტოკენის გენერაცია)

  მეთოდი: POST

  გამოძახება: api/authentication/authenticate

  Request Body:

```
{
   "login": "your login",
   "password": "your password"
}
```

  სადაც:

  login (string) - API -სთან დაშვების ლოგინი,

  password (string) - API -სთან დაშვების პაროლი

  Response:

```
{
   "token": "eyJhbGciOiJIUzUxMiIsInR5cCI6IkpXVCJ9……..0kJov3akIA",
   "ex": null
}
```

  სადაც:

  token (string) - წარმოადგენს დაგენერირებულ ტოკენს (მოქმედია 36 საათის განმავლობაში),

  ex (string) - ინფორმაცია შეცდომის შესახებ (ასეთის არსებობის შემთხვევაში)

getCustomersByCode - მყიდველები საიდენტ. კოდის მიხედვით

- **ფუნქცია:** getCustomersByCode

  აღწერა: მყიდველების წამოღება საიდენტიფიკაციო კოდის მიხედვით

  მეთოდი: GET

  გამოძახება: api/operation/getCustomersByCode/{code}

  სადაც: code (string) - წარმოადგენს მყიდველის საიდენტიფიკაციო კოდს.

  Response:


```
{
   "contragents": [{
      "id": 1,
      "group_id": 5,
      "code": "12345678910",
      "name": "სატესტო სატესტო",
      "address": "",
      "phone": "",
      "email": "",
      "is_company": true,
      "is_resident": true,
      "vat_type": 1,
      "cons_period": 30,
      "add_fields": [{
         "field": "usr_column_515",
         "value": ""
      }]
   }],
   "ex": null
}
```


  სადაც:

  contragents - მყიდველების კოლექციაა რომელიც შედგება:

  id (int) - მყიდველის Id,

  group_id (int) - მყიდველის ჯგუფის Id,

  code (string[50]) - მყიდველის საიდენტიფიკაციო კოდი ან პირადი ნომერი,

  name (string[200]) - დასახელება,

  address (string[200]) - მისამართი,

  tel (string[50]) - ტელეფონის ნომერი,

  mail (string[50]) - ელ. ფოსტა,

  is_company (bool) - მყიდველის ტიპი (true - იურიდიული პირი, false - ფიზიკური პირი).

is_resident (bool) - რეზიდენტი (true - ადგილობრივი, false - უცხო ქვეყნის მოქალაქე),

vat_type (byte) - დღგ-ს ტიპი (0 - არ არის დღგ-ს გადამხდელი, 1 - დღგ-ს გადამხდელი, 2 - განთავისუფლებული ჩათვლის უფლებით, 3 - განთავისუფლებული ჩათვლის უფლების გარეშე),

cons_period (int) - კონსიგნაციის ვადა (დღე),

add_fields - მყიდველის დამატებითი ველების კოლექცია სადაც:

field (string) - დამატებითი ველის დასახელება,

value (string) - დამატებითი ველის მნიშვნელობა.

ex (string) - ინფორმაცია შეცდომის შესახებ (ასეთის არსებობის შემთხვევაში).

getVendorsByCode - მომწოდებლები საიდენტ. კოდის მიხედვით

- **ფუნქცია:** getVendorsByCode

  აღწერა: მომწოდებლების წამოღება საიდენტიფიკაციო კოდის მიხედვით

  მეთოდი: GET

  გამოძახება: api/operation/getVendorsByCode/{code}

  სადაც: code (string) - წარმოადგენს მომწოდებლის საიდენტიფიკაციო კოდს.

  Response:


```
{
   "contragents": [{
      "id": 2,
      "group_id": 3,
      "code": "00112233665",
      "name": "X ვენდორ",
      "address": "",
      "phone": "",
      "email": "",
      "is_company": true,
      "is_resident": true,
      "vat_type": 1,
      "cons_period": 30,
      "add_fields": [{
         "field": "usr_column_515",
         "value": ""
      }]
   }],
   "ex": null
}
```


  სადაც:

  contragents - მომწოდებლების კოლექციაა რომელიც შედგება:

  id (int) - მომწოდებლის Id,

  group_id (int) - მომწოდებლის ჯგუფის Id,

  code (string[50]) - მომწოდებლის საიდენტიფიკაციო კოდი ან პირადი ნომერი,

  name (string[200]) - დასახელება,

  address (string[200]) - მისამართი,

  tel (string[50]) - ტელეფონის ნომერი,

  mail (string[50]) - ელ. ფოსტა,

  is_company (bool) - მომწოდებლის ტიპი (true - იურიდიული პირი, false - ფიზიკური პირი).

  is_resident (bool) - რეზიდენტი (true - ადგილობრივი, false - უცხო ქვეყნის მოქალაქე),

vat_type (byte) - დღგ-ს ტიპი (0 - არ არის დღგ-ს გადამხდელი, 1 - დღგ-ს გადამხდელი, 2 - განთავისუფლებული ჩათვლის უფლებით, 3 - განთავისუფლებული ჩათვლის უფლების გარეშე),

cons_period (int) - კონსიგნაციის ვადა (დღე),

add_fields - მომწოდებლის დამატებითი ველების კოლექცია სადაც:

field (string) - დამატებითი ველის დასახელება,

value (string) - დამატებითი ველის მნიშვნელობა.

ex (string) - ინფორმაცია შეცდომის შესახებ (ასეთის არსებობის შემთხვევაში).

getCustomers - მყიდველები

- **ფუნქცია:** getCustomers

  აღწერა: მყიდველების წამოღება

  მეთოდი: GET

  გამოძახება: api/operation/getCustomers

  Response:


```
{
   "contragents": [{
      "id": 1,
      "group_id": 5,
      "code": "12345678910",
      "name": "სატესტო სატესტო",
      "address": "",
      "phone": "",
      "email": "",
      "is_company": true,
      "is_resident": true,
      "vat_type": 1,
      "cons_period": 30,
      "add_fields": [{
         "field": "usr_column_515",
         "value": ""
      }]
   }],
   "ex": null
}
```


  სადაც:

  contragents - მყიდველების კოლექციაა რომელიც შედგება:

  id (int) - მყიდველის Id,

  group_id (int) - მყიდველის ჯგუფის Id,

  code (string[50]) - მყიდველის საიდენტიფიკაციო კოდი ან პირადი ნომერი,

  name (string[200]) - დასახელება,

  address (string[200]) - მისამართი,

  tel (string[50]) - ტელეფონის ნომერი,

  mail (string[50]) - ელ. ფოსტა,

  is_company (bool) - მყიდველის ტიპი (true - იურიდიული პირი, false - ფიზიკური პირი).

  is_resident (bool) - რეზიდენტი (true - ადგილობრივი, false - უცხო ქვეყნის მოქალაქე),

vat_type (byte) - დღგ-ს ტიპი (0 - არ არის დღგ-ს გადამხდელი, 1 - დღგ-ს გადამხდელი, 2 - განთავისუფლებული ჩათვლის უფლებით, 3 - განთავისუფლებული ჩათვლის უფლების გარეშე),

cons_period (int) - კონსიგნაციის ვადა (დღე),

add_fields - მყიდველის დამატებითი ველების კოლექცია სადაც:

field (string) - დამატებითი ველის დასახელება,

value (string) - დამატებითი ველის მნიშვნელობა.

ex (string) - ინფორმაცია შეცდომის შესახებ (ასეთის არსებობის შემთხვევაში).

getVendors - მომწოდებლები

- **ფუნქცია:** getVendors

  აღწერა: მომწოდებლების წამოღება

  მეთოდი: GET

  გამოძახება: api/operation/getVendors

Response:


```
{
   "contragents": [{
      "id": 2,
      "group_id": 3,
      "code": "00112233665",
      "name": "X ვენდორ",
      "address": "",
      "phone": "",
      "email": "",
      "is_company": true,
      "is_resident": true,
      "vat_type": 1,
      "cons_period": 30,
      "add_fields": [{
         "field": "usr_column_515",
         "value": ""
      }]
   }],
   "ex": null
}
```


სადაც:

contragents - მომწოდებლების კოლექციაა რომელიც შედგება:

id (int) - მომწოდებლის Id,

group_id (int) - მომწოდებლის ჯგუფის Id,

code (string[50]) - მომწოდებლის საიდენტიფიკაციო კოდი ან პირადი ნომერი,

name (string[200]) - დასახელება,

address (string[200]) - მისამართი,

tel (string[50]) - ტელეფონის ნომერი,

mail (string[50]) - ელ. ფოსტა,

is_company (bool) - მომწოდებლის ტიპი (true - იურიდიული პირი, false - ფიზიკური პირი).

is_resident (bool) - რეზიდენტი (true - ადგილობრივი, false - უცხო ქვეყნის მოქალაქე),

vat_type (byte) - დღგ-ს ტიპი (0 - არ არის დღგ-ს გადამხდელი, 1 - დღგ-ს გადამხდელი, 2 - განთავისუფლებული ჩათვლის უფლებით, 3 - განთავისუფლებული ჩათვლის უფლების გარეშე),

cons_period (int) - კონსიგნაციის ვადა (დღე),

add_fields - მომწოდებლის დამატებითი ველების კოლექცია სადაც:

field (string) - დამატებითი ველის დასახელება,

value (string) - დამატებითი ველის მნიშვნელობა.

ex (string) - ინფორმაცია შეცდომის შესახებ (ასეთის არსებობის შემთხვევაში).

getCustomerAdditionalFields - მყიდველის დამატებითი ველების აღწერა

- **ფუნქცია:** getCustomerAdditionalFields

  აღწერა: მყიდველის დამატებითი ველების აღწერის წამოღება

  მეთოდი: GET

  გამოძახება: api/operation/getCustomerAdditionalFields

  Response:

```
{
   "fields": [{
      "name": "usr_column_515",
      "header": "ლიმიტი"
   }],
   "ex": null
}
```

სადაც:

fields - მყიდველის დამატებითი ველების აღწერის კოლექციაა რომელიც შედგება:

name (string) - მყიდველის დამატებითი ველის დასახელება მონაცემთა ბაზაში (Column),

header (string) - მომხმარებლის მიერ მყიდველის დამატებითი ველისთვის დარქმეული სახელი.

ex (string) - ინფორმაცია შეცდომის შესახებ (ასეთის არსებობის შემთხვევაში).

getVendorAdditionalFields - მომწოდებლის დამატებითი ველების აღწერა

- **ფუნქცია:** getVendorAdditionalFields

  აღწერა: მომწოდებლის დამატებითი ველების აღწერის წამოღება

  მეთოდი: GET

  გამოძახება: api/operation/getVendorAdditionalFields

  Response:

```
{
   "fields": [{
      "name": "usr_column_515",
      "header": "ლიმიტი"
   }],
   "ex": null
}
```

სადაც:

fields - მომწოდებლის დამატებითი ველების აღწერის კოლექციაა რომელიც შედგება:

name (string) - მომწოდებლის დამატებითი ველის დასახელება მონაცემთა ბაზაში (Column),

header (string) - მომხმარებლის მიერ მომწოდებლის დამატებითი ველისთვის დარქმეული სახელი.

ex (string) - ინფორმაცია შეცდომის შესახებ (ასეთის არსებობის შემთხვევაში).

getCustomerGroups - მყიდველის ჯგუფები

- **ფუნქცია:** getCustomerGroups

  აღწერა: მყიდველის ჯგუფების წამოღება

  მეთოდი: GET

  გამოძახება: api/operation/getCustomerGroups

  Response:

```
{
   "groups": [{
      "id": 5,
      "parent_id": 2,
      "path": "0#2#5",
      "name": "მყიდველები"
   }, {
      "id": 22,
      "parent_id": 5,
      "path": "0#2#5#22",
      "name": "მყიდველები Level2"
   }],
   "ex": null
}
```

  სადაც:

  groups - მყიდველის ჯგუფების კოლექციაა რომელიც შედგება:

  id (int) - ჯგუფის Id,

  parent_id (int) - მშობელი ჯგუფის Id,

  path (string) - ჯგუფის ჩანაწერის მისამართი,

  name (string) - ჯგუფის დასახელება,

  ex (string) - ინფორმაცია შეცდომის შესახებ (ასეთის არსებობის შემთხვევაში).

getVendorGroups - მომწოდებლის ჯგუფები

- **ფუნქცია:** getVendorGroups

  აღწერა: მომწოდებლის ჯგუფების წამოღება

  მეთოდი: GET

  გამოძახება: api/operation/getVendorGroups

  Response:

```
{
   "groups": [{
      "id": 3,
      "parent_id": 1,
      "path": "0#1#3",
      "name": "მომწოდებლები"
   }, {
      "id": 23,
      "parent_id": 3,
      "path": "0#1#3#23",
      "name": "იმპორტიორები"
   }],
   "ex": null
}
```

  სადაც:

  groups - მომწოდებლის ჯგუფების კოლექციაა რომელიც შედგება:

  id (int) - ჯგუფის Id,

  parent_id (int) - მშობელი ჯგუფის Id,

  path (string) - ჯგუფის ჩანაწერის მისამართი,

  name (string) - ჯგუფის დასახელება,

  ex (string) - ინფორმაცია შეცდომის შესახებ (ასეთის არსებობის შემთხვევაში).

getProductGroups - საქონლის ჯგუფები

- **ფუნქცია:** getProductGroups

  აღწერა: საქონლის ჯგუფების წამოღება

  მეთოდი: GET

  გამოძახება: api/operation/getProductGroups

  Response:

```
{
   "groups": [{
      "id": 10,
      "parent_id": 1,
      "order_id": 10,
      "path": "0#1#10",
      "name": "სასაქონლო–მატერიალური აქტივები"
   }, {
      "id": 11,
      "parent_id": 10,
      "order_id": 11,
      "path": "0#1#10#11",
      "name": "საქონელი"
   }, {
      "id": 16,
      "parent_id": 10,
      "order_id": 16,
      "path": "0#1#10#16",
      "name": "შენიშვნები"
   }],
   "ex": null
}
```

  სადაც:

  groups - საქონლის ჯგუფების კოლექციაა რომელიც შედგება:

  id (int) - ჯგუფის Id,

  parent_id (int) - მშობელი ჯგუფის Id,

  order_id (int) - სორტირება,

  path (string) - ჯგუფის ჩანაწერის მისამართი,

  name (string) - ჯგუფის დასახელება,

  ex (string) - ინფორმაცია შეცდომის შესახებ (ასეთის არსებობის შემთხვევაში).

getWebProductGroups - საქონლის ალტერნატიული (ვებ) ჯგუფები

- **ფუნქცია:** getWebProductGroups

  აღწერა: საქონლის ალტერნატიული (ვებ) ჯგუფების წამოღება

  მეთოდი: GET

  გამოძახება: api/operation/getWebProductGroups

Response:


```
{
   "groups": [{
      "id": 10,
      "parent_id": 1,
      "order_id": 10,
      "path": "0#1#10",
      "image": null,
      "name": "სასაქონლო ჯგუფები",
      "name2": "",
      "name3": ""
   }, {
      "id": 12,
      "parent_id": 11,
      "order_id": 13,
      "path": "0#1#10#11#12",
      "image": null,
      "name": "ციტრუსი",
      "name2": "",
      "name3": ""
   }],
   "ex": null
}
```


სადაც:

groups - საქონლის ალტერნატიული ჯგუფების კოლექციაა რომელიც შედგება:

id (int) - ჯგუფის Id,

parent_id (int) - მშობელი ჯგუფის Id,

order_id (int) - სორტირება,

path (string) - ჯგუფის ჩანაწერის მისამართი,

image (string) - სურათი (base64 of byte[]) ,

name (string) - ჯგუფის დასახელება,

name2 (string) - ჯგუფის ალტერნატიული დასახელება,

name3 (string) - ჯგუფის ალტერნატიული დასახელება3,

ex (string) - ინფორმაცია შეცდომის შესახებ (ასეთის არსებობის შემთხვევაში).

getProvidedServiceGroups - გაწეული მომსახურების ჯგუფები

- **ფუნქცია:** getProvidedServiceGroups

  აღწერა: გაწეული მომსახურების ჯგუფების წამოღება

  მეთოდი: GET

  გამოძახება: api/operation/getProvidedServiceGroups

  Response:

```
{
   "groups": [{
      "id": 110,
      "parent_id": 2,
      "order_id": 110,
      "path": "0#2#110",
      "name": "გაწეული მომსახურება"
   }, {
      "id": 135,
      "parent_id": 110,
      "order_id": 135,
      "path": "0#2#110#135",
      "name": "ქვე ჯგუფი"
   }],
   "ex": null
}
```

სადაც:

groups - გაწეული მომსახურების ჯგუფების კოლექციაა რომელიც შედგება:

id (int) - ჯგუფის Id,

parent_id (int) - მშობელი ჯგუფის Id,

order_id (int) - სორტირება,

path (string) - ჯგუფის ჩანაწერის მისამართი,

name (string) - ჯგუფის დასახელება,

ex (string) - ინფორმაცია შეცდომის შესახებ (ასეთის არსებობის შემთხვევაში).

getReceivedServiceGroups - მიღებული მომსახურების ჯგუფები

- **ფუნქცია:** getReceivedServiceGroups

  აღწერა: მიღებული მომსახურების ჯგუფების წამოღება

  მეთოდი: GET

  გამოძახება: api/operation/getReceivedServiceGroups

  Response:

```
{
   "groups": [{
      "id": 120,
      "parent_id": 2,
      "order_id": 120,
      "path": "0#2#120",
      "name": "მიღებული მომსახურება"
   }, {
      "id": 121,
      "parent_id": 120,
      "order_id": 121,
      "path": "0#2#120#121",
      "name": "საიჯარო ქირა"
   }],
   "ex": null
}
```

სადაც:

groups - მიღებული მომსახურების ჯგუფების კოლექციაა რომელიც შედგება:

id (int) - ჯგუფის Id,

parent_id (int) - მშობელი ჯგუფის Id,

order_id (int) - სორტირება,

path (string) - ჯგუფის ჩანაწერის მისამართი,

name (string) - ჯგუფის დასახელება,

ex (string) - ინფორმაცია შეცდომის შესახებ (ასეთის არსებობის შემთხვევაში).

getInventoryGroups - ძირითადი საშუალებების ჯგუფები

- **ფუნქცია:** getInventoryGroups

  აღწერა: ძირითადი საშუალებების ჯგუფების წამოღება

  მეთოდი: GET

  გამოძახება: api/operation/getInventoryGroups

  Response:


```
{
   "groups": [{
      "id": 40,
      "parent_id": 3,
      "order_id": 40,
      "path": "0#3#40",
      "name": "ძირითადი საშუალებები"
   }, {
      "id": 41,
      "parent_id": 40,
      "order_id": 41,
      "path": "0#3#40#41",
      "name": "მიწის ნაკვეთები"
   }],
   "ex": null
}
```


  სადაც:

  groups - ძირითადი საშუალების ჯგუფების კოლექციაა რომელიც შედგება:

  id (int) - ჯგუფის Id,

  parent_id (int) - მშობელი ჯგუფის Id,

  order_id (int) - სორტირება,

  path (string) - ჯგუფის ჩანაწერის მისამართი,

  name (string) - ჯგუფის დასახელება,

  ex (string) - ინფორმაცია შეცდომის შესახებ (ასეთის არსებობის შემთხვევაში).

getProducts - საქონლის კატალოგი

- **ფუნქცია:** getProducts

  აღწერა: საქონლის კატალოგის წამოღება

  მეთოდი: GET

  გამოძახება: api/operation/getProducts

  Response:


```
{
   "products": [{
      "id": 5,
      "group_id": 11,
      "web_group_id": 18,
      "unit_id": 1,
      "code": "00011111111111111105",
      "name": "ანოს ასკილი",
      "name_eng": null,
      "name_rus": null,
      "comment": "",
      "partnumber": "",
      "weight": 0,
      "volume": 0,
      "vat": 1,
      "order_id": 1,
      "add_fields": [{
         "field": "usr_column_501",
         "value": ""
      }, {
         "field": "usr_column_502",
         "value": "უშაქრო"
      }, {
         "field": "usr_column_503",
         "value": ""
      }]
   }],
   "ex": null
}
```


  სადაც:

  products - საქონლის კატალოგის კოლექციაა რომელიც შედგება:

  id (int) - საქონლის Id,

  group_id (int) - საქონლის ჯგუფის Id,

  web_group_id (int) - საქონლის ალტერნატიული ჯგუფის Id,

  unit_id (int) - საქონლის ერთეულის Id,

code (string) - საქონლის კოდი,

name (string) - საქონლის დასახელება,

name_eng (string) - საქონლის დასახელება ინგლისურ ენაზე,

name_rus (string) - საქონლის დასახელება რუსულ ენაზე,

comment (string) - კომენტარი,

partnumber (string) - არტიკული,

weight (double) - წონა,

volume (double) - მოცულობა,

vat (byte) - დღგ-ს ტიპი (1 - იბეგრება, 2 - ნულოვანი, 3 - დაუბეგრავი),

order_id (int) - სორტირება,

add_fields - საქონლის დამატებითი ველების კოლექცია სადაც:

field (string) - დამატებითი ველის დასახელება,

value (string) - დამატებითი ველის მნიშვნელობა.

ex (string) - ინფორმაცია შეცდომის შესახებ (ასეთის არსებობის შემთხვევაში).

getProductsArray - შერჩეული საქონლის კატალოგი

- **ფუნქცია:** getProductsArray

  აღწერა: შერჩეული საქონლის კატალოგის წამოღება

  მეთოდი: POST

  გამოძახება: api/operation/getProductsArray

  Request Body:

  ```
  [
    1, 2
  ]
  ```

  სადაც:

  int[] - საქონლის Id -ების კოლექციაა

  Response:

დაბრუნებული საქონლის კატალოგის კოლექცია ანალოგიურია **getProducts** - მეთოდით დაბრუნებული კოლექციისა. (იმ შემთხვევაში თუ გადასაცემი პარამეტრის რაოდენობა აღემატება 2000 - ს, რეკომენდირებულია გამოიძახოთ **getProducts** მეთოდი)

getProductsAfter - საქონლის კატალოგი

- **ფუნქცია:** getProductsAfter

  აღწერა: შეცვლილი საქონლის კატალოგის წამოღება

  მეთოდი: GET

  გამოძახება: api/operation/getProductsAfter/{after_date}

  სადაც:

  after_date (datetime) - თარიღი (yyyy-MM-ddTHH:mm:ss), რომლის შემდგომი ცვლილებებიც გვაინტერესებს.

  Response:

  დაბრუნებული საქონლის კატალოგის კოლექცია ანალოგიურია **getProducts** - მეთოდით დაბრუნებული კოლექციისა.

getProvidedServices - გაწეული მომსახურების კატალოგი

- **ფუნქცია:** getProvidedServices

  აღწერა: გაწეული მომსახურების კატალოგის წამოღება

  მეთოდი: GET

  გამოძახება: api/operation/getProvidedServices

Response:


```
{
   "services": [{
      "id": 8,
      "group_id": 110,
      "unit_id": 17,
      "code": "02",
      "name": "მომსახურება",
      "comment": "",
      "vat": 1,
      "add_fields": []
   }, {
      "id": 11,
      "group_id": 110,
      "unit_id": 17,
      "code": "",
      "name": "Service",
      "comment": "",
      "vat": 1,
      "add_fields": []
   }],
   "ex": null
}
```


სადაც:

services - გაწეული მომსახურების კატალოგის კოლექციაა რომელიც შედგება:

id (int) - გაწეული მომსახურების Id,

group_id (int) - გაწეული მომსახურების ჯგუფის Id,

unit_id (int) - გაწეული მომსახურების ერთეულის Id,

code (string) - გაწეული მომსახურების კოდი,

name (string) - გაწეული მომსახურების დასახელება,

comment (string) - კომენტარი,

vat (byte) - დღგ-ს ტიპი (1 - იბეგრება, 2 - ნულოვანი, 3 - დაუბეგრავი),

add_fields - გაწეული მომსახურების დამატებითი ველების კოლექცია სადაც:

field (string) - დამატებითი ველის დასახელება,

value (string) - დამატებითი ველის მნიშვნელობა.

ex (string) - ინფორმაცია შეცდომის შესახებ (ასეთის არსებობის შემთხვევაში).

getReceivedServices - მიღებული მომსახურების კატალოგი

- **ფუნქცია:** getReceivedServices

  აღწერა: მიღებული მომსახურების კატალოგის წამოღება

  მეთოდი: GET

  გამოძახება: api/operation/getReceivedServices

Response:


```
{
   "services": [{
      "id": 7,
      "group_id": 132,
      "unit_id": 17,
      "code": "151515",
      "name": "მასაჟი",
      "comment": "",
      "vat": 1,
      "add_fields": [{
         "field": "usr_column_535",
         "value": "13"
      }]
   }, {
      "id": 17,
      "group_id": 129,
      "unit_id": 1,
      "code": "00012",
      "name": "ყავა",
      "comment": "",
      "vat": 1,
      "add_fields": [{
         "field": "usr_column_535",
         "value": ""
      }]
   }],
   "ex": null
}
```


სადაც:

services - მიღებული მომსახურების კატალოგის კოლექციაა რომელიც შედგება:

id (int) - მიღებული მომსახურების Id,

group_id (int) - მიღებული მომსახურების ჯგუფის Id,

unit_id (int) - მიღებული მომსახურების ერთეულის Id,

code (string) - მიღებული მომსახურების კოდი,

name (string) - მიღებული მომსახურების დასახელება,

comment (string) - კომენტარი,

vat (byte) - დღგ-ს ტიპი (1 - იბეგრება, 2 - ნულოვანი, 3 - დაუბეგრავი),

add_fields - მიღებული მომსახურების დამატებითი ველების კოლექცია სადაც:

field (string) - დამატებითი ველის დასახელება,

value (string) - დამატებითი ველის მნიშვნელობა.

ex (string) - ინფორმაცია შეცდომის შესახებ (ასეთის არსებობის შემთხვევაში).

getInventories - ძირითადი საშუალებების კატალოგი

- **ფუნქცია:** getInventories

  აღწერა: ძირითადი საშუალებების კატალოგის წამოღება

  მეთოდი: GET

  გამოძახება: api/operation/getInventories

Response:


```
{
   "inventories": [{
      "id": 14,
      "group_id": 41,
      "unit_id": 1,
      "code": "00009",
      "name": "inv1",
      "comment": "",
      "in_date": null,
      "add_fields": [{
         "field": "usr_column_532",
         "value": "30.01"
      }]
   }, {
      "id": 22,
      "group_id": 48,
      "unit_id": 1,
      "code": "00124",
      "name": "trans",
      "comment": "",
      "in_date": "2018-12-26T17:24:03",
      "add_fields": [{
         "field": "usr_column_532",
         "value": ""
      }]
   }],
   "ex": null
}
```


სადაც:

services - ძირითადი საშუალების კატალოგის კოლექციაა რომელიც შედგება:

id (int) - ძირითადი საშუალების Id,

group_id (int) - ძირითადი საშუალების ჯგუფის Id,

unit_id (int) - ძირითადი საშუალების ერთეულის Id,

code (string) - ძირითადი საშუალების კოდი,

name (string) - ძირითადი საშუალების დასახელება,

comment (string) - კომენტარი,

in_date (datetime?) - ექსპლუატაციაში შესვლის თარიღი,

add_fields - ძირითადი საშუალების დამატებითი ველების კოლექცია სადაც:

field (string) - დამატებითი ველის დასახელება,

value (string) - დამატებითი ველის მნიშვნელობა.

ex (string) - ინფორმაცია შეცდომის შესახებ (ასეთის არსებობის შემთხვევაში).

getProductAdditionalFields - პროდუქტის დამატებითი ველების აღწერა

- **ფუნქცია:** getProductAdditionalFields

  აღწერა: პროდუქტის დამატებითი ველების აღწერის წამოღება

  მეთოდი: GET

  გამოძახება: api/operation/getProductAdditionalFields

  Response:

```
{
   "fields": [{
      "name": "usr_column_501",
      "header": "ველი1"
   }, {
      "name": "usr_column_502",
      "header": "ველი2"
   }, {
      "name": "usr_column_503",
      "header": "მწარმოებელი კომპანია"
   }],
   "ex": null
}
```

სადაც:

fields - პროდუქტის დამატებითი ველების აღწერის კოლექციაა რომელიც შედგება:

name (string) - პროდუქტის დამატებითი ველის დასახელება მონაცემთა ბაზაში (Column),

header (string) - მომხმარებლის მიერ პროდუქტის დამატებითი ველისთვის დარქმეული სახელი.

ex (string) - ინფორმაცია შეცდომის შესახებ (ასეთის არსებობის შემთხვევაში).

getProvidedServiceAdditionalFields - გაწეული მომსახურების დამატებითი ველების აღწერა

- **ფუნქცია:** getProvidedServiceAdditionalFields

  აღწერა: გაწეული მომსახურების დამატებითი ველების აღწერის წამოღება

  მეთოდი: GET

  გამოძახება: api/operation/getProvidedServiceAdditionalFields

  Response:

```
{
   "fields": [{
      "name": "usr_column_506",
      "header": "About"
   }],
   "ex": null
}
```

სადაც:

fields - გაწეული მომსახურების დამატებითი ველების აღწერის კოლექციაა რომელიც შედგება:

name (string) - გაწეული მომსახურების დამატებითი ველის დასახელება მონაცემთა ბაზაში (Column),

header (string) - მომხმარებლის მიერ გაწეული მომსახურების დამატებითი ველისთვის დარქმეული სახელი.

ex (string) - ინფორმაცია შეცდომის შესახებ (ასეთის არსებობის შემთხვევაში).

getInventoryAdditionalFields - ძირითადი საშუალების დამატებითი ველების აღწერა

- **ფუნქცია:** getInventoryAdditionalFields

  აღწერა: ძირითადი საშუალების დამატებითი ველების აღწერის წამოღება

  მეთოდი: GET

  გამოძახება: api/operation/getInventoryAdditionalFields

  Response:

```
{
   "fields": [{
      "name": "usr_column_532",
      "header": "გახარჯვა"
   }],
   "ex": null
}
```

სადაც:

fields - ძირითადი საშუალების დამატებითი ველების აღწერის კოლექციაა რომელიც შედგება:

name (string) - ძირითადი საშუალების დამატებითი ველის დასახელება მონაცემთა ბაზაში (Column),

header (string) - მომხმარებლის მიერ ძირითადი საშუალების დამატებითი ველისთვის დარქმეული სახელი.

ex (string) - ინფორმაცია შეცდომის შესახებ (ასეთის არსებობის შემთხვევაში).

getCharacteristics - პროდუქტის მახასიათებლების აღწერა

- **ფუნქცია:** getCharacteristics

  აღწერა: პროდუქტის მახასიათებლების აღწერის წამოღება

  მეთოდი: GET

  გამოძახება: api/operation/getCharacteristics

  Response:

```
{
   "characteristics": [{
      "id": 1,
      "tag": "COLOR",
      "type": 0,
      "name": "ფერი",
      "name2": "Color",
      "name3": "Колір"
   }, {
      "id": 4,
      "tag": "POWER",
      "type": 0,
      "name": "სიმძლავრე",
      "name2": "Power",
      "name3": "потужність"
   }],
   "ex": null
}
```

სადაც:

characteristics - პროდუქტის მახასიათებლების აღწერის კოლექციაა რომელიც შედგება:

id (int) - მახასიათებლის id,

tag (string) - მახასიათებლის tag (უნიკალური ყოველი მახასიათებლისთვის),

type (byte) - მახასიათებლის ტიპი (0 - ტექსტური, 1 - სია),

name (string) - მახასიათებლის დასახელება,

name2 (string) - მახასიათებლის ალტერნატიული დასახელება,

name3 (string) - მახასიათებლის ალტერნატიული დასახელება,

ex (string) - ინფორმაცია შეცდომის შესახებ (ასეთის არსებობის შემთხვევაში).

getCharacteristicValues - პროდუქტის მახასიათებლები

- **ფუნქცია:** getCharacteristicValues

  აღწერა: პროდუქტის მახასიათებლების წამოღება

  მეთოდი: GET

  გამოძახება: api/operation/getCharacteristicValues

  Response:


```
{
   "characteristic_values": [{
      "product_id": 43,
      "characteristic_id": 1,
      "value": "შავი",
      "value2": "black",
      "value3": "чорний"
   }, {
      "product_id": 43,
      "characteristic_id": 4,
      "value": "18 ვატი",
      "value2": "18 watt",
      "value3": "18 ватт"
   }, {
      "product_id": 44,
      "characteristic_id": 1,
      "value": "წითელი",
      "value2": "red",
      "value3": "Червоний"
   }],
   "ex": null
}
```


სადაც:

characteristic_values - პროდუქტის მახასიათებლების კოლექციაა რომელიც შედგება:

product_id (int) - პროდუქტის id,

characteristic_id (int) - მახასიათებლის id,

value (string) - მახასიათებლის მნიშვნელობა მოცემული საქონლისთვის,

value2 (string) - მახასიათებლის ალტერნატიული მნიშვნელობა მოცემული საქონლისთვის,

value3 (string) - მახასიათებლის ალტერნატიული მნიშვნელობა მოცემული საქონლისთვის,

ex (string) - ინფორმაცია შეცდომის შესახებ (ასეთის არსებობის შემთხვევაში).

getCharacteristicValuesArray - შერჩეული პროდუქტის მახასიათებლები

- **ფუნქცია:** getCharacteristicValuesArray

  აღწერა: შერჩეული პროდუქტის მახასიათებლების წამოღება

  მეთოდი: POST

  გამოძახება: api/operation/getCharacteristicValuesArray

  Request Body:

  ```
  [
    43, 44
  ]
  ```

  სადაც:

  int[] - საქონლის Id -ების კოლექციაა.

  Response:

  დაბრუნებული პროდუქტის მახასიათებლების კოლექცია ანალოგიურია **getCharacteristicValues** - მეთოდით დაბრუნებული კოლექციისა.

getPackedProducts - ერთად დაჯგუფებული პროდუქტები

- **ფუნქცია:** getPackedProducts

  აღწერა: ერთად დაჯგუფებული პროდუქტების წამოღება

  მეთოდი: GET

  გამოძახება: api/operation/getPackedProducts

  Response:


```
{
   "pack_info": [{
      "product_id": 1,
      "pack_id": 1
   }, {
      "product_id": 2,
      "pack_id": 1
   }, {
      "product_id": 3,
      "pack_id": 1
   }, {
      "product_id": 4,
      "pack_id": 3
   }, {
      "product_id": 5,
      "pack_id": 3
   }],
   "ex": null
}
```


სადაც:

pack_info - ერთად დაჯგუფებული პროდუქტის კოლექციაა რომელიც შედგება:

product_id (int) - პროდუქტის id,

pack_id (int) - მაჯგუფებელი id,

ex (string) - ინფორმაცია შეცდომის შესახებ (ასეთის არსებობის შემთხვევაში).

getSubCodeTypes - ქვე კოდის ტიპები

- **ფუნქცია:** getSubCodeTypes

  აღწერა: ქვე კოდის ტიპების წამოღება

  მეთოდი: GET

  გამოძახება: api/operation/getSubCodeTypes

  Response:

```
{
   "types": [{
      "name": "sub_column_1",
      "header": "სერია"
   }, {
      "name": "sub_column_2",
      "header": "ვადა"
   }],
   "ex": null
}
```

სადაც:

types - ქვე კოდის ტიპების კოლექციაა რომელიც შედგება:

name (string) - ქვე კოდის ტიპის დასახელება მონაცემთა ბაზაში (Column),

header (string) - მომხმარებლის მიერ ქვე კოდის ტიპის დარქმეული სახელი.

ex (string) - ინფორმაცია შეცდომის შესახებ (ასეთის არსებობის შემთხვევაში).

getProductSubCodes - საქონლის ქვე კოდები

- **ფუნქცია:** getProductSubCodes

  აღწერა: საქონლის ქვე კოდების წამოღება

  მეთოდი: GET

  გამოძახება: api/operation/getProductSubCodes

  Response:


```
{
   "sub_codes": [{
      "product_id": 9,
      "sub_id": 1,
      "barcode": "",
      "sub_columns": [{
         "field": "sub_column_1",
         "value": "001"
      }, {
         "field": "sub_column_2",
         "value": "2018-09-04T00:00:00"
      }]
   }, {
      "product_id": 9,
      "sub_id": 2,
      "barcode": "",
      "sub_columns": [{
         "field": "sub_column_1",
         "value": "002"
      }, {
         "field": "sub_column_2",
         "value": "2019-09-04T00:00:00"
      }]
   }],
   "ex": null
}
```


სადაც:

sub_codes - საქონლის ქვე კოდების კოლექციაა რომელიც შედგება:

product_id (int) - საქონლის id,

sub_id (int) - საქონლის ქვე კოდის id,

barcode (string) - საქონლის ქვე კოდის შტრიხკოდი,

sub_columns - საქოლის ქვე კოდის ველების მნიშვნელობები რომელიც შედგება:

field (string) - საქონლის ქვე კოდის ველის დასახელება,

value (object) - საქონლის ქვე კოდის ველის მნიშვნელობა.

ex (string) - ინფორმაცია შეცდომის შესახებ (ასეთის არსებობის შემთხვევაში).

getProductPlaces - საქონლის შენახვის ადგილები

- **ფუნქცია:** getProductPlaces

  აღწერა: საქონლის შენახვის ადგილის წამოღება

  მეთოდი: GET

  გამოძახება: api/operation/getProductPlaces

  Response:

```
{
   "places": [{
      "product_id": 10,
      "store_id": 1,
      "place": "თარო#5"
   }, {
      "product_id": 10,
      "store_id": 2,
      "place": ""
   }],
   "ex": null
}
```

სადაც:

places - საქონლის შენახვის ადგილის კოლექციაა რომელიც შედგება:

product_id (int) - საქონლის id,

store_id (int) - საწყობის id,

place (string) - საქონლის შენახვის ადგილი შესაბამისი საწყობისთვის,

ex (string) - ინფორმაცია შეცდომის შესახებ (ასეთის არსებობის შემთხვევაში).

getProductImages - საქონლის სურათები

- **ფუნქცია:** getProductImages

  აღწერა: საქონლის სურათების წამოღება

  მეთოდი: GET

  გამოძახება: api/operation/getProductImages/{product}

  სადაც: product (int) - წარმოადგენს საქონლის Id-ს.

  Response:

```
{
   "images": [{
      "id": 1,
      "image": "image base64 string"
   }, {
      "id": 2,
      "image": "image base64 string2"
   }],
   "ex": null
}
```

სადაც:

images - საქონლის სურათების კოლექციაა რომელიც შედგება:

id (int) - საქონლის სურათის id,

image (string) - საქონლის სურათი (base64 of byte[]) ,

ex (string) - ინფორმაცია შეცდომის შესახებ (ასეთის არსებობის შემთხვევაში).

getProductsImageArray - შერჩეული საქონლის სურათები

- **ფუნქცია:** getProductsImageArray

  აღწერა: შერჩეული საქონლის სურათების წამოღება

  მეთოდი: POST

  გამოძახება: api/operation/getProductsImageArray

  Request Body:

```
[
   5, 9
]
```

  სადაც:

  int[] - საქონლის Id -ების კოლექციაა (რეკომენდირებულია მაქს: 20 ერთეული).

  Response:

```
{
   "images": [{
      "id": 26,
      "product_id": 5,
      "image": "image base64 string"
   }, {
      "id": 27,
      "product_id": 5,
      "image": "image base64 string"
   }, {
      "id": 29,
      "product_id": 5,
      "image": null
   }, {
      "id": 31,
      "product_id": 9,
      "image": "image base64 string"
   }],
   "ex": null
}
```

  სადაც:

  images - შერჩეული საქონლის სურათების კოლექციაა რომელიც შედგება:

  id (int) - საქონლის სურათის id,

  product_id (int) - საქონლის id,

  image (string) - საქონლის სურათი (base64 of byte[]) ,

  ex (string) - ინფორმაცია შეცდომის შესახებ (ასეთის არსებობის შემთხვევაში).

getProductsBarcodeArray - შერჩეული საქონლის შტრიხკოდები

- **ფუნქცია:** getProductsBarcodeArray

  აღწერა: შერჩეული საქონლის შტრიხკოდების წამოღება

  მეთოდი: POST

  გამოძახება: api/operation/getProductsBarcodeArray

  Request Body:

```
[
   1, 2, 3
]
```

  სადაც:

  int[] - საქონლის Id -ების კოლექციაა.

  Response:

```
{
   "barcodes": [{
      "product_id": 3,
      "barcode": "34712051673775"
   }, {
      "product_id": 2,
      "barcode": "67327054128376"
   }, {
      "product_id": 1,
      "barcode": "1122"
   }, {
      "product_id": 1,
      "barcode": " 3333"
   }],
   "ex": null
}
```

  სადაც:

  barcodes - შერჩეული საქონლის შტრიხკოდების კოლექციაა რომელიც შედგება:

  product_id (int) - საქონლის id,

  barcode (string) - საქონლის შტრიხკოდი ,

  ex (string) - ინფორმაცია შეცდომის შესახებ (ასეთის არსებობის შემთხვევაში).

getProductsOnWay - საქონლის მომლოდინე შეკვეთები (საქონელი გზაში)

- **ფუნქცია:** getProductsOnWay

  აღწერა: საქონლის მომლოდინე შეკვეთების (საქონელი გზაში) წამოღება

  მეთოდი: GET

  გამოძახება: api/operation/getProductsOnWay

  Response:

```
{
   "on_way": [{
      "id": 10,
      "date": "2018-09-19",
      "quantity": 10
   }],
   "ex": null
}
```

სადაც:

on_way - საქონლის მომლოდინე შეკვეთების კოლექციაა რომელიც შედგება:

id (int) - საქონლის id,

date (date) - ჩამოსვლის თარიღი,

quantity (decimal) - მომლოდინე საქონლის რაოდენობა,

ex (string) - ინფორმაცია შეცდომის შესახებ (ასეთის არსებობის შემთხვევაში).

getProductPrices - საქონლის ფასები

- **ფუნქცია:** getProductPrices

  აღწერა: საქონლის ფასების წამოღება

  მეთოდი: GET

  გამოძახება: api/operation/getProductPrices

  Response:


```
{
   "prices": [{
      "product_id": 1,
      "price_id": 3,
      "price": 20.0,
      "discount_price": 7.0,
      "discount_start": "2018-12-03T00:00:00",
      "discount_end": "2018-12-05T00:00:00"
   }, {
      "product_id": 1,
      "price_id": 4,
      "price": 1.0,
      "discount_price": 0.0,
      "discount_start": null,
      "discount_end": null
   }, {
      "product_id": 2,
      "price_id": 4,
      "price": 0.0,
      "discount_price": 0.0,
      "discount_start": null,
      "discount_end": null
   }],
   "ex": null
}
```


  სადაც:

  prices - საქონლის ფასების კოლექციაა რომელიც შედგება:

  product_id (int) - საქონლის id,

  price_id (int) - ფასის ტიპის id,

  price (double) - საქონლის ფასი,

  discount_price (double) - საქონლის ფადაკლებული ფასი,

  discount_start (datetime) - ფასდაკლების დაწყების თარიღი,

  discount_end (datetime) - ფასდაკლების დასრულების თარიღი,

  ex (string) - ინფორმაცია შეცდომის შესახებ (ასეთის არსებობის შემთხვევაში).

getProductPricesAdvance - მითითებული საქონლის ფასები

- **ფუნქცია:** getProductPricesAdvance

  აღწერა: საქონლის ფასების წამოღება მითითებული ფილტრით

  მეთოდი: POST

  გამოძახება: api/operation/getProductPricesAdvance

  Request Body:

```
{
   "prods": [1, 2],
   "price": 3
}
```

  სადაც:

  prods (int[]) - საქონლის Id -ების კოლექციაა,

  price (int) - ფასის ტიპის Id (Default -3 (საცალო ფასი)).

  Response:

```
{
   "prices": [{
      "product_id": 1,
      "price": 20.0,
      "discount_price": 7.0,
      "discount_start": "2018-12-03T00:00:00",
      "discount_end": "2018-12-05T00:00:00"
   }, {
      "product_id": 2,
      "price": 0.0,
      "discount_price": 0.0,
      "discount_start": null,
      "discount_end": null
   }],
   "ex": null
}
```

  სადაც:

  prices - საქონლის ფასების კოლექციაა რომელიც შედგება:

  product_id (int) - საქონლის id,

  price (double) - საქონლის ფასი,

  discount_price (double) - საქონლის ფადაკლებული ფასი,

  discount_start (datetime) - ფასდაკლების დაწყების თარიღი,

  discount_end (datetime) - ფასდაკლების დასრულების თარიღი,

  ex (string) - ინფორმაცია შეცდომის შესახებ (ასეთის არსებობის შემთხვევაში).

getProductsRest - საქონლის მიმდინარე ნაშთი თითოეული საწყობისთვის

- **ფუნქცია:** getProductsRest

  აღწერა: საქონლის მიმდინარე ნაშთის წამოღება (თითოეული საწყობისთვის)

  მეთოდი: GET

  გამოძახება: api/operation/getProductsRest

  Response:

```
{
   "rest": [{
      "id": 1,
      "store": 1,
      "rest": 0,
      "reserve": 2
   }, {
      "id": 1,
      "store": 2,
      "rest": 1,
      "reserve": 0
   }],
   "ex": null
}
```

  სადაც:

  rest - საქონლის ნაშთის კოლექციაა რომელიც შედგება:

  id (int) - საქონლის Id,

  store (int) - საწყობის Id.

  rest (decimal) - საქონლის ნაშთი (რეზერვის გათვალისწინებით).

  reserve (decimal) - საქონლის რეზერვი.

  ex (string) - ინფორმაცია შეცდომის შესახებ (ასეთის არსებობის შემთხვევაში).

getProductsRestArray - შერჩეული საქონლის მიმდინარე ნაშთი თითოეული საწყობისთვის

- **ფუნქცია:** getProductsRestArray

  აღწერა: შერჩეული საქონლის მიმდინარე ნაშთის წამოღება (თითოეული საწყობისთვის)

  მეთოდი: POST

  გამოძახება: api/operation/getProductsRestArray

  Request Body:

```
{
   "prods": [1, 2]
}
```

  სადაც:

  prods (int[]) - საქონლის Id -ების კოლექციაა

  Response:

დაბრუნებული საქონლის ნაშთის კოლექცია ანალოგიურია **getProductsRest** - მეთოდით დაბრუნებული კოლექციისა.

getSubProductsRest - საქონლის ქვე კოდების მიმდინარე ნაშთი თითოეული საწყობისთვის

- **ფუნქცია:** getSubProductsRest

  აღწერა: საქონლის ქვე კოდების მიმდინარე ნაშთის წამოღება (თითოეული საწყობისთვის)

  მეთოდი: GET

  გამოძახება: api/operation/getSubProductsRest

  Response:

```
{
   "rest": [{
      "id": 9,
      "sub_id": 1,
      "store": 1,
      "rest": 6,
      "reserve": 0
   }, {
      "id": 9,
      "sub_id": 1,
      "store": 2,
      "rest": 0,
      "reserve": 0
   }],
   "ex": null
}
```

სადაც:

rest - საქონლის ნაშთების კოლექციაა რომელიც შედგება:

id (int) - საქონლის Id,

sub_id (int) - საქონლის ქვე კოდის Id.

store (int) - საწყობის Id.

rest (decimal) - საქონლის ქვე კოდის ნაშთი (რეზერვის გათვალისწინებით).

reserve (decimal) - საქონლის ქვე კოდის რეზერვი.

ex (string) - ინფორმაცია შეცდომის შესახებ (ასეთის არსებობის შემთხვევაში).

getProductsRestByStore - საქონლის მიმდინარე ნაშთი კონკრეტული საწყობისთვის

- **ფუნქცია:** getProductsRestByStore

  აღწერა: საქონლის მიმდინარე ნაშთის წამოღება (კონკრეტული საწყობისთვის)

  მეთოდი: GET

  გამოძახება: api/operation/getProductsRestByStore/{store}

  სადაც: store (int) - წარმოადგენს საწყობის Id-ს.

  Response:

```
{
   "store_rest": [{
      "id": 1,
      "rest": 0
   }, {
      "id": 2,
      "rest": 1
   }],
   "ex": null
}
```

  სადაც:

  store_rest - საქონლის ნაშთის კოლექციაა რომელიც შედგება:

  id (int) - საქონლის Id,

  rest (decimal) - საქონლის ნაშთი მითითებულ საწყობში.

  ex (string) - ინფორმაცია შეცდომის შესახებ (ასეთის არსებობის შემთხვევაში).

getProductsRestAdvance - საქონლის მიმდინარე ნაშთი ფასით

- **ფუნქცია:** getProductsRestAdvance

  აღწერა: საქონლის მიმდინარე ნაშთის და ფასის წამოღება (კონკრეტული საწყობისთვის)

  მეთოდი: POST

  გამოძახება: api/operation/getProductsRestAdvance

  Request Body:

```
{
   "prods": [1, 2],
   "store": 1,
   "price": 3
}
```

  სადაც:

  prods (int[]) - საქონლის Id -ების კოლექციაა,

  store (int) - საწყობის Id (0 - ყველა საწყობში ერთად აღებული),

  price (int) - ფასის ტიპის Id (Default -3 (საცალო ფასი)).

  Response:

```
{
   "rest_info": [{
      "id": 1,
      "rest": 0,
      "price": 7.25
   }, {
      "id": 2,
      "rest": 1,
      "price": 2.5
   }],
   "ex": null
}
```

  სადაც:

  rest_info - საქონლის ნაშთის კოლექციაა რომელიც შედგება:

  id (int) - საქონლის Id,

  rest (decimal) - საქონლის ნაშთი მითითებულ საწყობში.

  price (double) - საქონლის გასაყიდი ფასი.

  ex (string) - ინფორმაცია შეცდომის შესახებ (ასეთის არსებობის შემთხვევაში).

getProductsSelfCost - საქონლის თვითღირებულება

- **ფუნქცია:** getProductsSelfCost

  აღწერა: საქონლის თვითღირებულების (AVERAGE) წამოღება

  მეთოდი: POST

  გამოძახება: api/operation/getProductsSelfCost

  Request Body:

```
{
   "prods": [1, 2],
   "date": "2018-11-15T23:59:59"
}
```

  სადაც:

  prods (int[]) - საქონლის Id -ების კოლექციაა,

  date (datetime) - თარიღი, რა დროისთვისაც ხდება თვითღირებულების გამოთვლა.

  Response:

```
{
   "cost_info": [{
      "id": 1,
      "cost": 32.751412429378533
   }, {
      "id": 2,
      "cost": 4.0
   }],
   "ex": null
}
```

  სადაც:

  cost_info - საქონლის თვითღირებულების კოლექციაა რომელიც შედგება:

  id (int) - საქონლის Id,

  cost (double) - საქონლის თვითღირებულება.

  ex (string) - ინფორმაცია შეცდომის შესახებ (ასეთის არსებობის შემთხვევაში).

getPriceTypes - ფასის ტიპები

- **ფუნქცია:** getPriceTypes

  აღწერა: ფასის ტიპების წამოღება

  მეთოდი: GET

  გამოძახება: api/operation/getPriceTypes

  Response:

```
{
   "types": [{
      "id": 3,
      "name": "საცალო"
   }, {
      "id": 4,
      "name": "მცირე საბითუმო"
   }, {
      "id": 5,
      "name": "საბითუმოX"
   }],
   "ex": null
}
```

  სადაც:

  types - ფასის ტიპების კოლექციაა რომელიც შედგება:

  id (int) - ფასის ტიპის Id,

  name (string) - ფასის ტიპის დასახელება.

  ex (string) - ინფორმაცია შეცდომის შესახებ (ასეთის არსებობის შემთხვევაში).

getDiscountTypes - ფასდაკლების სახეობები

- **ფუნქცია:** getDiscountTypes

  აღწერა: ფასდაკლების სახეობების წამოღება

  მეთოდი: GET

  გამოძახება: api/operation/getDiscountTypes

  Response:

```
{
   "types": [{
      "id": 1,
      "discount_percent": 15.2
   }, {
      "id": 2,
      "discount_percent": 10.0
   }],
   "ex": null
}
```

  სადაც:

  types - ფასდაკლების სახეობების კოლექციაა რომელიც შედგება:

  id (int) - ფასდაკლების სახეობის Id,

  discount_percent (double) - ფასდაკლების მნიშვნელობა(%).

  ex (string) - ინფორმაცია შეცდომის შესახებ (ასეთის არსებობის შემთხვევაში).

getUnits - საზომი ერთეულები

- **ფუნქცია:** getUnits

  აღწერა: საზომი ერთეულების წამოღება

  მეთოდი: GET

  გამოძახება: api/operation/getUnits

  Response:

```
{
   "units": [{
      "id": 1,
      "name": "ც",
      "full_name": "ცალი"
   }, {
      "id": 2,
      "name": "კგ",
      "full_name": "კილოგრამი"
   }],
   "ex": null
}
```

  სადაც:

  units - საზომი ერთეულების კოლექციაა რომელიც შედგება:

  id (int) - ერთეულის Id,

  name (string) - ერთეულის მოკლე დასახელება,

  full_name (string) - ერთეულის სრული დასახელება.

  ex (string) - ინფორმაცია შეცდომის შესახებ (ასეთის არსებობის შემთხვევაში).

getStoreGroups - საწყობის ჯგუფები

- **ფუნქცია:** getStoreGroups

  აღწერა: საწყობის ჯგუფების წამოღება

  მეთოდი: GET

  გამოძახება: api/operation/getStoreGroups

  Response:

```
{
   "groups": [{
      "id": 2,
      "parent_id": 1,
      "path": "0#1#2",
      "name": "საწყობები"
   }, {
      "id": 3,
      "parent_id": 1,
      "path": "0#1#3",
      "name": "მაღაზიები"
   }, {
      "id": 4,
      "parent_id": 1,
      "path": "0#1#4",
      "name": "ბორტები"
   }],
   "ex": null
}
```

  სადაც:

  groups - საწყობის ჯგუფების კოლექციაა რომელიც შედგება:

  id (int) - ჯგუფის Id,

  parent_id (int) - მშობელი ჯგუფის Id,

  path (string) - ჯგუფის ჩანაწერის მისამართი,

  name (string) - ჯგუფის დასახელება.

  ex (string) - ინფორმაცია შეცდომის შესახებ (ასეთის არსებობის შემთხვევაში).

getStores - საწყობები

- **ფუნქცია:** getStores

  აღწერა: საწყობების წამოღება

  მეთოდი: GET

  გამოძახება: api/operation/getStores

  Response:


```
{
   "stores": [{
      "id": 1,
      "group_id": 2,
      "name": "მთავარი საწყობი",
      "address": "",
      "project_id": 1
   }, {
      "id": 2,
      "group_id": 2,
      "name": "საწყობი 2",
      "address": "",
      "project_id": 2
   }],
   "ex": null
}
```


  სადაც:

  stores - საწყობების კოლექციაა რომელიც შედგება:

  id (int) - საწყობის Id,

  group_id (int) - საწყობის ჯგუფის Id,

  name (string) - საწყობის დასახელება,

  address (string) - საწყობის მისამართი,

  project_id (int) - პროექტის Id.

  ex (string) - ინფორმაცია შეცდომის შესახებ (ასეთის არსებობის შემთხვევაში).

getProjects - პროექტები

- **ფუნქცია:** getProjects

  აღწერა: პროექტების წამოღება

  მეთოდი: GET

  გამოძახება: api/operation/getProjects

  Response:

```
{
   "projects": [{
      "id": 1,
      "name": "ძირითადი პროექტი"
   }, {
      "id": 2,
      "name": "პროექტი2"
   }, {
      "id": 3,
      "name": "პროექტი3"
   }],
   "ex": null
}
```

  სადაც:

  projects - პროექტების კოლექციაა რომელიც შედგება:

  id (int) - პროექტის Id,

  name (string) - პროექტის დასახელება,

  ex (string) - ინფორმაცია შეცდომის შესახებ (ასეთის არსებობის შემთხვევაში).

getTerminals - POS ტერმინალები

- **ფუნქცია:** getTerminals

  აღწერა: POS ტერმინალების წამოღება

  მეთოდი: GET

  გამოძახება: api/operation/getTerminals

  Response:

```
{
   "terminals": [{
      "id": 10,
      "name": "POS 1"
   }, {
      "id": 14,
      "name": "POS 2"
   }],
   "ex": null
}
```

  სადაც:

  terminals - POS ტერმინალების კოლექციაა რომელიც შედგება:

  id (int) - POS ტერმინალის Id,

  name (string) - POS ტერმინალის დასახელება,

  ex (string) - ინფორმაცია შეცდომის შესახებ (ასეთის არსებობის შემთხვევაში).

getCashes - სალაროები

- **ფუნქცია:** getCashes

  აღწერა: სალაროების წამოღება

  მეთოდი: GET

  გამოძახება: api/operation/getCashes

  Response:

```
{
   "cashes": [{
      "id": 1,
      "name": "მთავარი სალარო"
   }, {
      "id": 2,
      "name": "სალარო #2"
   }],
   "ex": null
}
```

  სადაც:

  cashes - სალაროების კოლექციაა რომელიც შედგება:

  id (int) - სალაროს Id,

  name (string) - სალაროს დასახელება,

  ex (string) - ინფორმაცია შეცდომის შესახებ (ასეთის არსებობის შემთხვევაში).

getUsers - მომხმარებლები

- **ფუნქცია:** getUsers

  აღწერა: მომხმარებლების წამოღება

  მეთოდი: GET

  გამოძახება: api/operation/getUsers

  Response:


```
{
   "users": [{
      "id": 1,
      "name": "ადმინისტრატორი ",
      "type": 1
   }, {
      "id": 2,
      "name": "სატესტო მოლარე",
      "type": 3
   }, {
      "id": 3,
      "name": "ოპერატორი ოპერატორი",
      "type": 2
   }],
   "ex": null
}
```


სადაც:

users - მომხმარებლების კოლექციაა რომელიც შედგება:

id (int) - მომხმარებლის Id,

name (string) - მომხმარებლის სახელი, გვარი

type (byte) - მომხმარებლის ტიპი (1 - ადმინისტრატორი, 2 - ოპერატორი, 3 - მოლარე ოპერატორი)

ex (string) - ინფორმაცია შეცდომის შესახებ (ასეთის არსებობის შემთხვევაში).

getUserPermissions - მომხმარებლის პრივილეგიები

- **ფუნქცია:** getUserPermissions

  აღწერა: მომხმარებლის პრივილეგიების წამოღება

  მეთოდი: GET

  გამოძახება: api/operation/getUserPermissions/{user}

  სადაც: user (int) - წარმოადგენს მომხმარებლის Id-ს.

  Response:

```
{
   "permissions": {
      "default_store": 1,
      "default_cash": 1,
      "default_price": 3,
      "max_discount": 10.0,
      "max_moneyin": 50000.0,
      "fiscal_print": 1,
      "stores": [1],
      "cashes": [1],
      "price_types": [3],
      "users": [1, 2, 3]
   },
   "ex": null
}
```

  სადაც:

  permissions - მომხმარებლის პრივილეგიების კოლექციაა რომელიც შედგება:

  default_store (int) - საწყობის Id (default საწყობი მოცემული მომხმარებლისთვის),

  default_ cash (int) - სალაროს Id (default სალარო მოცემული მომხმარებლისთვის),

  default_ price (int) - ფასის ტიპის Id (default ფასის ტიპი მოცემული მომხმარებლისთვის),

  max_discount (decimal) - ფასდაკლების მაქსიმალურად დასაშვები მნიშვნელობა (%) ,

  max_moneyin (decimal) - მიღებული თანხის მაქსიმალურად დასაშვები მნიშვნელობა ,

  fiscal_print (byte) - ფისკალური ჩეკის ამობეჭდვის რეჟიმი (0 - არ დაიბეჭდოს, 1 - დეტალური, 2 - ჯამური),

  stores (int[]) - საწყობების კოლექცია რომელთან წვდომაც დაშვებულია მოცემული მომხმარებლისთვის,

  cashes (int[]) - სალაროების კოლექცია რომელთან წვდომაც დაშვებულია მოცემული მომხმარებლისთვის,

  price_types (int[]) - ფასის ტიპების კოლექცია რომელთან წვდომაც დაშვებულია მოცემული მომხმარებლისთვის (1 - მიღების ფასი, 2- თვითღირებულება, ყველა სხვა ციფრი რომელიც >=3 - შესაბამისი ფასის ტიპის Id -ა).

users (int[]) - მომხმარებლების კოლექცია რომელთა გატარებულ ოპერაციებზეც წვდომა დაშვებულია მოცემული მომხმარებლისთვის (ReadOnly).

ex (string) - ინფორმაცია შეცდომის შესახებ (ასეთის არსებობის შემთხვევაში).

getBankAccounts - საბანკო ანგარიშები

- **ფუნქცია:** getBankAccounts

  აღწერა: საბანკო ანგარიშების წამოღება

  მეთოდი: GET

  გამოძახება: api/operation/getBankAccounts

  Response:

```
{
   "accounts": [{
      "id": 1,
      "code": "TBCBGE22",
      "name": "თიბისი ბანკი",
      "account": "GE218541545456554551",
      "currency": "GEL"
   }, {
      "id": 2,
      "code": "TBCBGE22",
      "name": "თიბისი ბანკი",
      "account": "GE218541545456554551",
      "currency": "USD"
   }],
   "ex": null
}
```

  სადაც:

  accounts - საბანკო ანგარიშების კოლექციაა რომელიც შედგება:

  id (int) - ანგარიშის Id,

  code (string) - ბანკის კოდი

  name (string) - ბანკის დასახელება,

  account (string) - ანგარიში,

  currency (string) - ანგარიშის ვალუტა

  ex (string) - ინფორმაცია შეცდომის შესახებ (ასეთის არსებობის შემთხვევაში).

getCreditBanks - განვადების ბანკები

- **ფუნქცია:** getCreditBanks

  აღწერა: განვადების ბანკების წამოღება

  მეთოდი: GET

  გამოძახება: api/operation/getCreditBanks

  Response:

```
{
   "credits": [{
      "id": 11,
      "name": "PCBG განვადება"
   }],
   "ex": null
}
```

  სადაც:

  accounts - განვადების ბანკების კოლექციაა რომელიც შედგება:

  id (int) - განვადების ბანკის Id,

  name (string) - განვადების ბანკის დასახელება.

  ex (string) - ინფორმაცია შეცდომის შესახებ (ასეთის არსებობის შემთხვევაში).

getStaffGroups - თანამშრომლის ჯგუფები

- **ფუნქცია:** getStaffGroups

  აღწერა: თანამშრომლის ჯგუფების წამოღება

  მეთოდი: GET

  გამოძახება: api/operation/getStaffGroups

  Response:

```
{
   "groups": [{
      "id": 2,
      "parent_id": 1,
      "path": "0#1#2",
      "name": "თანამშრომლები"
   }, {
      "id": 3,
      "parent_id": 2,
      "path": "0#1#2#3",
      "name": "level1"
   }],
   "ex": null
}
```

  სადაც:

  groups - თანამშრომლის ჯგუფების კოლექციაა რომელიც შედგება:

  id (int) - ჯგუფის Id,

  parent_id (int) - მშობელი ჯგუფის Id,

  path (string) - ჯგუფის ჩანაწერის მისამართი,

  name (string) - ჯგუფის დასახელება.

  ex (string) - ინფორმაცია შეცდომის შესახებ (ასეთის არსებობის შემთხვევაში).

getStaffs - თანამშრომლები

- **ფუნქცია:** getStaffs

  აღწერა: თანამშრომლების წამოღება

  მეთოდი: GET

  გამოძახება: api/operation/getStaffs

  Response:


```
{
   "staffs": [{
      "id": 1,
      "group_id": 2,
      "name": "თანამშრომელი1",
      "private_num": "11111111111",
      "passport_num": "xx123465",
      "address": "მისამართი",
      "tel": "+99500000000",
      "comment": "დამატებითი ინფორმაცია",
      "add_fields": [{
         "field": "usr_column_507",
         "value": "30"
      }]
   }, {
      "id": 5,
      "group_id": 2,
      "name": "თანამშრომელი2",
      "private_num": "11111111112",
      "passport_num": "xx1234652",
      "address": "მისამართი2",
      "tel": "+99500000002",
      "comment": "დამატებითი ინფორმაცია2",
      "add_fields": [{
         "field": "usr_column_507",
         "value": "30"
      }]
   }],
   "ex": null
}
```


  სადაც:

  staffs - თანამშრომლების კოლექციაა რომელიც შედგება:

  id (int) - თანამშრომლის Id,

  group_id (int) - თანამშრომლის ჯგუფის Id,

  name (string) - თანამშრომლის დასახელება,

private_num (string) - თანამშრომლის პირადი ნომერი,

passport_num (string) - თანამშრომლის პასპორტის ნომერი,

address (string) - თანამშრომლის მისამართი,

tel (string) - თანამშრომლის ტელეფონი,

comment (string) - კომენტარი,

add_fields - თანამშრომლის დამატებითი ველების კოლექცია სადაც:

field (string) - დამატებითი ველის დასახელება,

value (string) - დამატებითი ველის მნიშვნელობა.

ex (string) - ინფორმაცია შეცდომის შესახებ (ასეთის არსებობის შემთხვევაში).

getStaffAdditionalFields - თანამშრომლის დამატებითი ველების აღწერა

- **ფუნქცია:** getStaffAdditionalFields

  აღწერა: თანამშრომლის დამატებითი ველების აღწერის წამოღება

  მეთოდი: GET

  გამოძახება: api/operation/getStaffAdditionalFields

  Response:

```
{
   "fields": [{
      "name": "usr_column_507",
      "header": "ასაკი"
   }],
   "ex": null
}
```

სადაც:

fields - თანამშრომლის დამატებითი ველების აღწერის კოლექციაა რომელიც შედგება:

name (string) - თანამშრომლის დამატებითი ველის დასახელება მონაცემთა ბაზაში (Column),

header (string) - თანამშრომლის მიერ მყიდველის დამატებითი ველისთვის დარქმეული სახელი.

ex (string) - ინფორმაცია შეცდომის შესახებ (ასეთის არსებობის შემთხვევაში).

getGiftCards - სასაჩუქრე ბარათები

- **ფუნქცია:** getGiftCards

  აღწერა: სასაჩუქრე ბარათების წამოღება

  მეთოდი: GET

  გამოძახება: api/operation/getGiftCards

  Response:


```
{
   "gifts": [{
      "id": 513449,
      "store": 27,
      "code": "0067631",
      "acc": "3121",
      "issuance_date": "2017-01-01 11:50:17",
      "amount": 200.00,
      "pay_amount": 200.00
   }, {
      "id": 513451,
      "store": 2,
      "code": "0069342",
      "acc": "3121",
      "issuance_date": "2017-01-01 13:35:27",
      "amount": 200.00,
      "pay_amount": 200.00
   }],
   "ex": null
}
```


  სადაც:

  gifts - სასაჩუქრე ბარათების კოლექციაა რომელიც შედგება:

  id (int) - ბარათის Id,

  store (int) - გამცემი მაღაზიის (საწყობის) Id,

  code (string) - ბარათის კოდი,

  acc (string) - ბარათის ბუღალტრული ანგარიში,

  issuance_date (DateTime) - გაცემის თარიღი,

  amount (decimal) - სასაჩუქრე ბარათის ღირებულება,

  pay_amount (decimal) - რეალურად გადახდილი თანხა,

  ex (string) - ინფორმაცია შეცდომის შესახებ (ასეთის არსებობის შემთხვევაში).

getBonusCoeff - ბონუსის კოეფიციენტი

- **ფუნქცია:** getBonusCoeff

  აღწერა: ბონუს კოეფიციენტის წამოღება

  მეთოდი: GET

  გამოძახება: api/operation/getBonusCoeff

  Response:

```
{
   "coeff": 1.8,
   "ex": null
}
```

  სადაც:

  coeff (double) - წარმოადგენს ბონუს კოეფიციენტს,

  ex (string) - ინფორმაცია შეცდომის შესახებ (ასეთის არსებობის შემთხვევაში).

getLoyaltyCardsByHolder - ლოიალობის ბარათები

- **ფუნქცია:** getLoyaltyCardsByHolder

  აღწერა: ლოიალობის ბარათების (აქტიური) წამოღება მფლობელის ს/კ მიხედვით.

  მეთოდი: GET

  გამოძახება: api/operation/getLoyaltyCardsByHolder/{holder_code}

  სადაც: holder_code (string) - წარმოადგენს მფლობელის საიდენტიფიკაციო კოდს.

  Response:

```
{
   "cards": [{
      "id": 2013,
      "holder_id": 41,
      "type_id": 1,
      "discount_id": 0,
      "price_id": 0,
      "code": "BC123",
      "info_code": "",
      "info_name": "",
      "info_address": "",
      "info_tel": ""
   }, {
      "id": 2014,
      "holder_id": 41,
      "type_id": 2,
      "discount_id": 2,
      "price_id": 0,
      "code": "BC123",
      "info_code": "",
      "info_name": "",
      "info_address": "",
      "info_tel": ""
   }],
   "ex": null
}
```

  სადაც:

  cards - ლოიალობის ბარათების კოლექციაა რომელიც შედგება:

  id (int) - ბარათის Id,

  holder_id (int) - ბარათის მფლობელის (კონტრაგენტის) Id,

  type_id (int) - ბარათის ტიპი (1 - დაგროვება, 2 - ფასდაკლება, 5 - ფასის ტიპი, 7 - cashback),

  discount_id (int) - ფასდაკლების Id (იმ შემთხვევაში როცა type_id = 2),

  price_id (int) - ფასის ტიპის Id (იმ შემთხვევაში როცა type_id = 5),

  code (string) - ბარათის კოდი,

info_code (string) - დამატებითი ინფორმაცია ბარათის მფლობელის კოდზე,

info_name (string) - დამატებითი ინფორმაცია ბარათის მფლობელის სახელზე,

info_address (string) - დამატებითი ინფორმაცია ბარათის მფლობელის მისამართზე,

info_tel (string) - დამატებითი ინფორმაცია ბარათის მფლობელის ტელეფონზე;

ex (string) - ინფორმაცია შეცდომის შესახებ (ასეთის არსებობის შემთხვევაში).

getBonusCardRestByCode - ბონუს ბარათის ნაშთი

- **ფუნქცია:** getBonusCardRestByCode

  აღწერა: ბონუს ბარათზე არსებული ნაშთის წამოღება ბარათის კოდის მიხედვით.

  მეთოდი: GET

  გამოძახება: api/operation/getBonusCardRestByCode/{code}

  სადაც: code (string) - წარმოადგენს ბარათის კოდს.

  Response:

  Response:

```
{
   "rest": 17.0,
   "ex": null
}
```

  სადაც:

  rest (decimal) - წარმოადგენს ბარათის ნაშთს,

  ex (string) - ინფორმაცია შეცდომის შესახებ (ასეთის არსებობის შემთხვევაში).

getAccountValue - ბუღალტრული ანგარიშის მნიშვნელობა

- **ფუნქცია:** getAccountValue

აღწერა: ბუღალტრული ანგარიშის ბრუნვის ჯამური მნიშვნელობის (დებეტ, კრედიტი) წამოღება

მეთოდი: POST

გამოძახება: api/operation/getAccountValue

Request Body:

```
{
   "primary_acc": "1410",
   "secondary_acc": "",
   "date": "2019-09-12T13:00:00",
   "obj": 6,
   "is_gel": true
}
```

სადაც:

primary_acc (string) - ანგარიშის კოდი,

secondary_acc (string) - ანგარიშის კოდი (საავანსო),

date (DateTime) - თარიღი,

obj (int) - იმ კონკრეტული ობიექტის Id, ვისი ანგარიშიცაა გადაცემული primary_acc ან secondary_acc - ში (მაგ: 1410 ის შემთხვევაში obj - იქნება კონრკეტული მყიდველის Id),

is_gel (bool) - შედეგი აჩვენოს ლარის ექვივალენტით.

Response:

```
{
   "values": {
      "debit_val": 1320.00,
      "credit_val": 0.00
   },
   "ex": null
}
```

სადაც:

values - შესაბამის ბუღალტრულ ანგარიშზე არსებული ბრუნვის ჯამური მნიშვნელობაა რომელიც შედგება :

debit_val (decimal) - დებეტ მნიშვნელობა,

credit_val (decimal) -) - კრედიტ მნიშვნელობა,

ex (string) - ინფორმაცია შეცდომის შესახებ (ასეთის არსებობის შემთხვევაში).

saveCustomer - მყიდველის შენახვა

- **ფუნქცია:** saveCustomer

  აღწერა: მყიდველის შენახვა (insert, update)

  მეთოდი: POST

  გამოძახება: api/operation/saveCustomer

  Request Body:

```
{
   "id": 0,
   "code": "12345678910",
   "name": "name1",
   "group_id": 5,
   "address": "Address1",
   "phone": "+99555000000",
   "email": "test@test.com",
   "vat_type": 1,
   "is_resident": false,
   "is_company": true,
   "cons_period": 30,
   "add_fields": [{
      "field": "usr_column_515",
      "value": "20"
   }]
}
```

  სადაც:

  id (int) - მყიდველის Id. (თუ იქმნება ახალი, გადაეცემა 0)

  code (string[50]) - მყიდველის საიდენტიფიკაციო კოდი ან პირადი ნომერი,

  name (string[200]) - დასახელება,

  group_id (int) - მყიდველის ჯგუფის Id. (სტანდარტულად = 5)

  address (string[200]) - მისამართი,

  phone (string[50]) - ტელეფონის ნომერი,

  email (string[50]) - ელ. ფოსტა,

  vat_type (byte) - დღგ-ს ტიპი (0 - არ არის დღგ-ს გადამხდელი, 1 - დღგ-ს გადამხდელი, 2 - განთავისუფლებული ჩათვლის უფლებით, 3 - განთავისუფლებული ჩათვლის უფლების გარეშე),

  is_resident (bool) - რეზიდენტი (true - ადგილობრივი, false - უცხო ქვეყნის მოქალაქე),

  is_company (bool) - მყიდველის ტიპი (true - იურიდიული პირი, false - ფიზიკური პირი),

  cons_period (int) - კონსიგნაციის პერიოდი (დღე),

  add_fields - მყიდველის დამატებითი ველების კოლექცია სადაც:

  field (string) - დამატებითი ველის დასახელება,

value (string) - დამატებითი ველის მნიშვნელობა.

Response:

```
{
   "id": 15,
   "ex": null
}
```

სადაც:

id (int) - დამატებული (ან დარედაქტირებული) მყიდველის Id

ex (string) - ინფორმაცია შეცდომის შესახებ (ასეთის არსებობის შემთხვევაში).

saveVendor - მომწოდებლის შენახვა

- **ფუნქცია:** saveVendor

  აღწერა: მომწოდებლის შენახვა (insert, update)

  მეთოდი: POST

  გამოძახება: api/operation/saveVendor

  Request Body:

```
{
   "id": 0,
   "code": "21345678910",
   "name": "name2",
   "group_id": 3,
   "address": "Address2",
   "phone": "+99555000001",
   "email": "test2@test.com",
   "vat_type": 1,
   "is_resident": false,
   "is_company": true,
   "cons_period": 10,
   "add_fields": [{
      "field": "usr_column_515",
      "value": "20"
   }]
}
```

  სადაც:

  id (int) - მომწოდებლის Id. (თუ იქმნება ახალი, გადაეცემა 0)

  code (string[50]) - მომწოდებლის საიდენტიფიკაციო კოდი ან პირადი ნომერი,

  name (string[200]) - დასახელება,

  group_id (int) - მომწოდებლის ჯგუფის Id. (სტანდარტულად = 3)

  address (string[200]) - მისამართი,

  phone (string[50]) - ტელეფონის ნომერი,

  email (string[50]) - ელ. ფოსტა,

  vat_type (byte) - დღგ-ს ტიპი (0 - არ არის დღგ-ს გადამხდელი, 1 - დღგ-ს გადამხდელი, 2 - განთავისუფლებული ჩათვლის უფლებით, 3 - განთავისუფლებული ჩათვლის უფლების გარეშე),

  is_resident (bool) - რეზიდენტი (true - ადგილობრივი, false - უცხო ქვეყნის მოქალაქე),

  is_company (bool) - მომწოდებლის ტიპი (true - იურიდიული პირი, false - ფიზიკური პირი),

  cons_period (int) - კონსიგნაციის პერიოდი (დღე),

  add_fields - მომწოდებლის დამატებითი ველების კოლექცია სადაც:

  field (string) - დამატებითი ველის დასახელება,

value (string) - დამატებითი ველის მნიშვნელობა.

Response:


```
{
   "id": 16,
   "ex": null
}
```


სადაც:

id (int) - დამატებული (ან დარედაქტირებული) მომწოდებლის Id

ex (string) - ინფორმაცია შეცდომის შესახებ (ასეთის არსებობის შემთხვევაში).

saveStaff - თანამშრომლის შენახვა

- **ფუნქცია:** saveStaff

  აღწერა: თანამშრომლის შენახვა (insert, update)

  მეთოდი: POST

  გამოძახება: api/operation/saveStaff

  Request Body:

```
{
    "id": 0,
    "group_id": 2,
    "name": "თანამშრომელი X",
    "private_num": "1122332211",
    "address": "",
    "tel": "+995551000",
    "comment": "კომენტარი",
    "add_fields": [{
       "field": "usr_column_508",
       "value": "დამატებით ველი2 "
    }]
 }
```

  სადაც:

  id (int) - თანამშრომლის Id. (თუ იქმნება ახალი, გადაეცემა 0),

  group_id (int) - თანამშრომლის ჯგუფის Id. (სტანდარტულად = 2),

  name (string[200]) - დასახელება,

  private_num (string[50]) - პირადი ნომერი,

  address (string[200]) - მისამართი,

  tel (string[200]) - ტელეფონი,

  comment (string[200]) - კომენტარი,

  add_fields - თანამშრომლის დამატებითი ველების კოლექცია სადაც:

  field (string) - დამატებითი ველის დასახელება,

  value (string) - დამატებითი ველის მნიშვნელობა.

  Response:

```
{
   "id": 1,
   "ex": null
}
```

სადაც:

id (int) - დამატებული (ან დარედაქტირებული) თანამშრომლის Id

ex (string) - ინფორმაცია შეცდომის შესახებ (ასეთის არსებობის შემთხვევაში).

saveProduct - საქონლის შენახვა

- **ფუნქცია:** saveProduct

  აღწერა: საქონლის შენახვა (insert, update)

  მეთოდი: POST

  გამოძახება: api/operation/saveProduct

  Request Body:

```
{
   "id": 0,
   "code": "123456-1",
   "name": "საქონელი X",
   "name_eng": "",
   "name_rus": "",
   "group_id": 11,
   "comment": "",
   "vat": 1,
   "plu": 0,
   "unit": 1,
   "prices": [{
      "id": 3,
      "value": 10
   }],
   "barcodes": [
      "QK124",
      "QK125"
   ],
   "images": ["image base64 string"],
   "add_fields": [{
      "field": "usr_column_501",
      "value": "თეთრი"
   }, {
      "field": "usr_column_502",
      "value": "XXL"
   }]
}
```

  სადაც:

  id (int) - საქონლის Id. (თუ იქმნება ახალი, გადაეცემა 0),

  code (string[50]) - საქონლის კოდი,

  name (string[200]) - დასახელება,

  name_eng (string[200]) - დასახელება ინგლისურ ენაზე,

  name_rus (string[200]) - დასახელება რუსულ ენაზე,

  group_id (int) - საქონლის ჯგუფის Id. (სტანდარტულად = 11),

comment (string[500]) - კომენტარი,

vat (byte) - დღგ-ს ტიპი (1 - იბეგრება, 2 - ნულოვანი, 3 - დაუბეგრავი),

plu (int) - plu კოდი წონითი საქონლისთვის > 0,

unit (int) - ერთეულის Id,

prices - კოლექცია შედგება:

id (int) - ფასის Id,

value (double) - ფასის მნიშვნელობა,

barcodes (string[]) შტრიხკოდების კოლექცია,

images (string[]) სურათების კოლექცია (base64 string ფორმატში).

add_fields - საქონლის დამატებითი ველების კოლექცია სადაც:

field (string) - დამატებითი ველის დასახელება,

value (string) - დამატებითი ველის მნიშვნელობა

(თუ prices კოლექცია არ არის null მოხდება მოცემული საქონლისთვის გადმოცემული ფაისის ტიპებისთვის შესაბამისი მნიშვნელობების მინიჭება).

(თუ barcodes კოლექცია არ არის null მოხდება მოცემული საქონლისთვის არსებული შტრიხკოდების წაშლა და გადმოცემული მნიშვნელობების შენახვა).

(თუ images კოლექცია არ არის null მოხდება მოცემული საქონლისთვის არსებული სურათების წაშლა და გადმოცემული მნიშვნელობების შენახვა).

Response:

```
{
   "id": 1,
   "ex": null
}
```

სადაც:

id (int) - დამატებული (ან დარედაქტირებული) საქონლის Id

ex (string) - ინფორმაცია შეცდომის შესახებ (ასეთის არსებობის შემთხვევაში).

saveProvidedService - გაწეული მომსახურების შენახვა

- **ფუნქცია:** saveProvidedService

  აღწერა: გაწეული მომსახურების შენახვა (insert, update)

  მეთოდი: POST

  გამოძახება: api/operation/saveProvidedService

  Request Body:

```
{
   "id": 0,
   "code": "123",
   "name": "მომსახურება X",
   "group_id": 110,
   "comment": "",
   "vat": 1,
   "unit": 1,
   "prices": [{
      "id": 3,
      "value": 100
   }],
   "add_fields": [{
      "field": "usr_column_506",
      "value": "XYZ"
   }]
}
```

  სადაც:

  id (int) - საქონლის Id. (თუ იქმნება ახალი, გადაეცემა 0),

  code (string[50]) - გაწეული მომსახურების კოდი,

  name (string[200]) - დასახელება,

  group_id (int) - გაწეული მომსახურების ჯგუფის Id. (სტანდარტულად = 110),

  comment (string[500]) - კომენტარი,

  vat (byte) - დღგ-ს ტიპი (1 - იბეგრება, 2 - ნულოვანი, 3 - დაუბეგრავი),

  unit (int) - ერთეულის Id,

  prices - კოლექცია შედგება:

  id (int) - ფასის Id,

  value (double) - ფასის მნიშვნელობა,

  add_fields - საქონლის დამატებითი ველების კოლექცია სადაც:

  field (string) - დამატებითი ველის დასახელება,

  value (string) - დამატებითი ველის მნიშვნელობა

(თუ prices კოლექცია არ არის null მოხდება მოცემული გაწეული მომსახურებისთვის გადმოცემული ფაისის ტიპებისთვის შესაბამისი მნიშვნელობების მინიჭება).

Response:

```
{
   "id": 2,
   "ex": null
}
```

სადაც:

id (int) - დამატებული (ან დარედაქტირებული) გაწეული მომსახურების Id,

ex (string) - ინფორმაცია შეცდომის შესახებ (ასეთის არსებობის შემთხვევაში).

getDocTypes - ოპერაციების სია

- **ფუნქცია:** getDocTypes

  აღწერა: FINA ში არსებული ოპერაციების აღწერის წამოღება

  მეთოდი: GET

  გამოძახება: api/operation/getDocTypes

  Response:

```
{
   "doc_types": [{
      "type": 1,
      "name": "ავანსის გადახურვა",
      "api_supported": false
   }, {
      "type": 8,
      "name": "შეკვეთა მყიდველისგან",
      "api_supported": true
   }],
   "ex": null
}
```

  სადაც:

  doc_types - FINA ში არსებული ოპერაციების აღწერის კოლექციაა რომელიც შედგება:

  type (int) - დოკუმენტის (ოპერაციის) ტიპი,

  name (string) - დოკუმენტის (ოპერაციის) დასახელება,

  api_supported (bool) - სტატუსი (მხარდაჭერილია თუ არა API ში)

  ex (string) - ინფორმაცია შეცდომის შესახებ (ასეთის არსებობის შემთხვევაში).

getDocAdditionalFields - დოკუმენტის დამატებითი ველების აღწერა

- **ფუნქცია:** getDocAdditionalFields

  აღწერა: დოკუმენტის დამატებითი ველების აღწერის წამოღება

  მეთოდი: GET

  გამოძახება: api/operation/getDocAdditionalFields/{type}

  სადაც: type (int) - წარმოადგენს შესაბამის დოკუმენტის ტიპს

  Response:

```
{
   "fields": [{
      "name": "usr_column_511",
      "header": "ლოკაცია"
   }],
   "ex": null
}
```

  სადაც:

  fields - შესაბამისი დოკუმენტის დამატებითი ველების აღწერის კოლექციაა რომელიც შედგება:

  name (string) - დოკუმენტის დამატებითი ველის დასახელება მონაცემთა ბაზაში (Column),

  header (string) - მომხმარებლის მიერ დამატებითი ველისთვის დარქმეული სახელი.

  ex (string) - ინფორმაცია შეცდომის შესახებ (ასეთის არსებობის შემთხვევაში).

getDocDiscountCard - გაცემული ფასდაკლების ბარათი

- **ფუნქცია:** getDocDiscountCard

  აღწერა: გაცემული ფასდაკლების ბარათის წამოღება

  მეთოდი: GET

  გამოძახება: api/operation/getDocDiscountCard/{id}

  სადაც: id (int) - წარმოადგენს შესაბამის ოპერაციის Id-ს.

  Response:

```
{
   "discount_card": {
      "id": 8733,
      "date": "2019-11-08T18:00:00",
      "num_pfx": "",
      "num": 15,
      "purpose": "ფასდაკლების ბარათის გაცემა",
      "amount": 823.44,
      "store": 1,
      "customer": 31,
      "user": 1,
      "staff": 0,
      "card_code": "2311",
      "discount_id": 2,
      "status": true,
      "ref_operatons": [8737, 8738]
   },
   "ex": null
}
```

  სადაც:

  discount_card - გაცემული ფასდაკლების ბარათია რომელიც შედგება:

  id (int) - ოპერაციის (ბარათის) id,

  date (datetime) - ოპერაციის თარიღი,

  num_pfx (string) - დოკუმენტის ნომრის პრეფიქსი,

  num (long) - დოკუმენტის ნომერი,

  purpose (string) - ოპერაციის შინაარსი,

  amount (decimal) - მყიდველის მიერ დაგროვილი თანხა (ტრანსაზციების თანხის ჯამი, რომელშიც აღნიშნული ბარათი მონაწილეობდა),

  store_id (int) - საწყობის id რომლიდანაც მოხდა ბარათის გაცემა,

  customer_id (int) - მყიდველის id,

  user (int) - მომხმარებლის (შემქმნელი) Id,

staff_id (int) - თანამშრომლის id,

card_code (string) - ბარათის კოდი,

discount_id (int) - ფასდაკლების id,

status (bool) - ბარათის სტატუსი (აქტიურია თუ არა),

ref_operatons (int[]) - ოპერაციები რომელშიც მონაწილეობდა აღნიშნული ბარათი.

ex (string) - ინფორმაცია შეცდომის შესახებ (ასეთის არსებობის შემთხვევაში).

getDocCustomerOrder - მყიდველისგან მიღებული შეკვეთა

- **ფუნქცია:** getDocCustomerOrder

  აღწერა: მყიდველისგან მიღებული შეკვეთის წამოღება

  მეთოდი: GET

  გამოძახება: api/operation/getDocCustomerOrder/{id}

  სადაც: id (int) - წარმოადგენს შესაბამისი ოპერაციის Id-ს.

  Response:


```
{
   "customer_order": {
      "id": 4165,
      "date": "2018-12-03T11:50:45.517",
      "num_pfx": "t",
      "num": 8,
      "purpose": "შეკვეთა მყიდველისგან № 8",
      "amount": 4.58,
      "currency": "USD",
      "rate": 2.6744,
      "store": 1,
      "customer": 1,
      "user": 1,
      "staff": 3,
      "project": 1,
      "is_vat": true,
      "pay_type": 1,
      "tr_start": "",
      "tr_end": "",
      "reserved": true,
      "actived": true,
      "add_fields": [{
         "field": "usr_column_511",
         "value": "41.710456, 44.769520"
      }],
      "products": [{
         "id": 2,
         "sub_id": 1,
         "price": 0.93,
         "quantity": 2
      }, {
         "id": 1,
         "sub_id": 0,
         "price": 2.71,
         "quantity": 1
      }],
      "services": []
```

```
    },
    "ex": null
}
```

სადაც:

customer_order - მყიდველისგან მიღებული შეკვეთაა რომელიც შედგება:

id (int) - დოკუმენტის id,

date (DateTime) - ოპერაციის თარიღი,

num_pfx (string) - დოკუმენტის ნომრის პრეფიქსი,

num (long) - დოკუმენტის ნომერი,

purpose (string) - დოკუმენტის შინაარსი (კომენტარი),

amount (decimal) - ოპერაციის ჯამური ღირებულება,

currency (string) - ვალუტის კოდი,

rate (decimal) - ვალუტის კურსი,

store (int) - საწყობის Id,

customer (int) - მყიდველის Id,

user (int) - მომხმარებლის (შემქმნელი) Id,

staff (int) - თანამშრომლის Id,

project (int) - პროექტის Id,

is_vat (bool) - დღგ-ს შეიცავს თუ არა,

pay_type (byte) - გადახდის ტიპი (0 - ნაღდი, 1 - უნაღდო, 2 - კონსიგნაცია, 3 - განვადება, 4 - ნაღდი/უნაღდო, 5 - უსასყიდლო, 6 - სხვა),

tr_start (string) - ტრანსპორტირების დაწყების ადგილი,

tr_end (string) - ტრანსპორტირების დასრულების ადგილი,

reserved (bool) - რეზერვაციის სტატუსი.

actived (bool) - შეკვეთის სტატუსი (აქტიურია თუ არა).

add_fields - დამატებითი ველების კოლექცია სადაც:

field (string) - დამატებითი ველის დასახელება,

value (string) - დამატებითი ველის მნიშვნელობა,

products - კოლექცია შედგება:

id (int) - საქონლის Id,

sub_id (int) - საქონლის ქვე-კოდის Id,

quantity (decimal) - საქონლის რაოდენობა,

price (decimal) - საქონლის ერთეულის ფასი,

services - კოლექცია შედგება:

id (int) - გაწეული მომსახურების Id,

quantity (decimal) - გაწეული მომსახურების რაოდენობა,

price (decimal) - გაწეული მომსახურების ერთეულის ფასი.

ex (string) - ინფორმაცია შეცდომის შესახებ (ასეთის არსებობის შემთხვევაში).

getDocProductOut - საქონლის რეალიზაცია

- **ფუნქცია:** getDocProductOut

  აღწერა: საქონლის რეალიზაციის წამოღება

  მეთოდი: GET

  გამოძახება: api/operation/getDocProductOut/{id}

  სადაც: id (int) - წარმოადგენს შესაბამისი ოპერაციის Id-ს.

  Response:


```
{
   "product_out": {
      "id": 6563,
      "date": "2019-05-16T15:28:19",
      "num_pfx": "",
      "num": 8624,
      "waybill_num": "0418925004",
      "purpose": "საქონლის  გაყიდვა №",
      "amount": 780.0,
      "currency": "GEL",
      "rate": 1.0,
      "store": 2,
      "customer": 1,
      "user": 1,
      "staff": 1,
      "project": 2,
      "is_vat": true,
      "make_entry": true,
      "pay_type": 2,
      "w_type": 2,
      "t_type": 1,
      "t_payer": 1,
      "w_cost": 0.0,
      "foreign": false,
      "drv_name": "რამინ ხოზრევანიძე",
      "tr_start": "ქ.ქუთაისი / \"+1\"ის ბაზარი eee",
      "tr_end": "წერეტლიე4",
      "driver_id": "61006038420",
      "car_num": "iu100iu",
      "tr_text": "",
      "overlap_type": 0,
      "overlap_amount": 0.0,
      "add_fields": [{
         "field": "usr_column_510",
         "value": ""
      }, {
         "field": "usr_column_527",
```

```
            "value": ""
        }, {
            "field": "usr_column_528",
            "value": ""
        }],
        "products": [{
            "id": 29,
            "sub_id": 1,
            "price": 4.5,
            "quantity": 120.0
        }, {
            "id": 28,
            "sub_id": 0,
            "price": 20.0,
            "quantity": 12.0
        }],
        "services": []
    },
    "ex": null
}
```


სადაც:

product_out - საქონლის რეალიზაციაა რომელიც შედგება:

id (int) - დოკუმენტის id,

date (DateTime) - ოპერაციის თარიღი,

num_pfx (string[20]) - დოკუმენტის ნომრის პრეფიქსი,

num (long) - დოკუმენტის ნომერი,

waybill_num (string) - ზედნადების ნომერი (RS),

purpose (string) - დოკუმენტის შინაარსი (კომენტარი),

amount (decimal) - ოპერაციის ჯამური ღირებულება,

currency (string) - ვალუტის კოდი,

rate (decimal) - ვალუტის კურსი,

store (int) - საწყობის Id,

customer (int) - მყიდველის Id,

user (int) - მომხმარებლის (შემქმნელი) Id,

staff (int) - თანამშრომლის Id,

project (int) - პროექტის Id,

is_vat (bool) - დღგ-ს შეიცავს თუ არა,

make_entry (bool) - ბუღალტრული გატარების სტატუსი,

pay_type (byte) - გადახდის ტიპი (0 - ნაღდი, 1 - უნაღდო, 2 - კონსიგნაცია, 3 - განვადება, 4 - ნაღდი/უნაღდო, 5 - უსასყიდლო, 6 - სხვა),

w_type (byte) - ზედნადების ტიპი (2 - ტრანსპორტირებით, 3 - ტრანსპორტირების გარეშე),

t_type (byte) - ტრანსპორტირების ტიპი (1 - საავტომობილო, 2 - სარკინიგზო, 3 - საავიაციო, 4 - სხვა, 6 - საავტომობილო უცხო ქვეყნის, 7 - გადამზიდავი საავტომობილო),

t_payer (byte) - ტრანსპორტირების ღირებულების გადამხდელი (1 - მყიდველი, 2 - გამყიდველი),

w_cost (decimal) - ტრანსპორტირების ხარჯი,

foreign (bool) - უცხო ქვეყნის მოქალაქე,

drv_name (string) - მძღოლის სახელი, გვარი,

tr_start (string) - ტრანსპორტირების დაწყების ადგილი,

tr_end (string) - ტრანსპორტირების დასრულების ადგილი,

driver_id (string) - მძღოლის პირადი ნომერი,

car_num (string) - ავტომობილის ნომერი,

tr_text (string) - მისაბმელი/გადამზიდავი (თუ ტრანსპორტირების ტიპი არის 7, გადამზიდავის საიდენტიფიკაციო კოდი, თუ ტრანსპორტირების ტიპი არის 4 - ტრანსპორტირების ფორმა, ხოლო საავტომობილოს შემთხვევაში - მისაბმელის ნომერი ასეთის არსებობის შემთხვევაში),

overlap_type (byte) - ავანსი/დღგ-ს გადახურვა (0 - არ გადაიხუროს არც ერთი, 1 - ავანსი გადაიხუროს, დღგ - ნაწილობრივ, 2 - ავანსი გადაიხუროს, დღგ - სრულად, 3 - ავანსი გადაიხუროს, დღგ - არა) ,

overlap_amount (decimal) - გადახურული დღგ- ს თანხა ,

add_fields - დამატებითი ველების კოლექცია სადაც:

field (string) - დამატებითი ველის დასახელება,

value (string) - დამატებითი ველის მნიშვნელობა,

products - კოლექცია შედგება:

id (int) - საქონლის Id,

sub_id (int) - საქონლის ქვე-კოდის Id,

quantity (decimal) - საქონლის რაოდენობა,

price (decimal) - საქონლის ერთეულის ფასი,

services - კოლექცია შედგება:

id (int) - გაწეული მომსახურების Id,

quantity (decimal) - გაწეული მომსახურების რაოდენობა,

price (decimal) - გაწეული მომსახურების ერთეულის ფასი.

ex (string) - ინფორმაცია შეცდომის შესახებ (ასეთის არსებობის შემთხვევაში).

getDocProductOutSingle - საცალო რეალიზაცია

- **ფუნქცია:** getDocProductOutSingle

  აღწერა: საქონლის საცალო რეალიზაციის წამოღება

  მეთოდი: GET

  გამოძახება: api/operation/getDocProductOutSingle/{id}

  სადაც: id (int) - წარმოადგენს შესაბამისი ოპერაციის Id-ს.

  Response:


```
{
   "product_out_single": {
      "id": 16902,
      "date": "2020-08-26T23:59:58",
      "num_pfx": "",
      "num": 20,
      "purpose": "დაჯგუფებული საცალო გაყიდვა staff1",
      "amount": 11.0,
      "currency": "GEL",
      "rate": 1.0,
      "store": 1,
      "user": 1,
      "staff": 2,
      "project": 1,
      "is_vat": true,
      "make_entry": true,
      "pay_type": 6,
      "add_fields": [],
      "products": [{
         "id": 1,
         "sub_id": 0,
         "price": 5.0,
         "quantity": 1.0
      }, {
         "id": 1,
         "sub_id": 0,
         "price": 6.0,
         "quantity": 1.0
      }]
   },
   "ex": null
}
```


  სადაც:

  product_out_single - საქონლის საცალო რეალიზაციაა რომელიც შედგება:

  id (int) - დოკუმენტის id,

date (DateTime) - ოპერაციის თარიღი,

num_pfx (string[20]) - დოკუმენტის ნომრის პრეფიქსი,

num (long) - დოკუმენტის ნომერი,

purpose (string) - დოკუმენტის შინაარსი (კომენტარი),

amount (decimal) - ოპერაციის ჯამური ღირებულება,

currency (string) - ვალუტის კოდი,

rate (decimal) - ვალუტის კურსი,

store (int) - საწყობის Id,

user (int) - მომხმარებლის (შემქმნელი) Id,

staff (int) - თანამშრომლის Id,

project (int) - პროექტის Id,

is_vat (bool) - დღგ-ს შეიცავს თუ არა,

make_entry (bool) - ბუღალტრული გატარების სტატუსი,

pay_type (byte) - გადახდის ტიპი (0 - ნაღდი, 1 - უნაღდო, 6 - სხვა),

add_fields - დამატებითი ველების კოლექცია სადაც:

field (string) - დამატებითი ველის დასახელება,

value (string) - დამატებითი ველის მნიშვნელობა,

products - კოლექცია შედგება:

id (int) - საქონლის Id,

sub_id (int) - საქონლის ქვე-კოდის Id,

quantity (decimal) - საქონლის რაოდენობა,

price (decimal) - საქონლის ერთეულის ფასი.

ex (string) - ინფორმაცია შეცდომის შესახებ (ასეთის არსებობის შემთხვევაში).

getDocInventoryOut - ძირითადი საშუალების რეალიზაცია

- **ფუნქცია:** getDocInventoryOut

  აღწერა: ძირითადი საშუალების რეალიზაციის წამოღება

  მეთოდი: GET

  გამოძახება: api/operation/getDocInventoryOut/{id}

  სადაც: id (int) - წარმოადგენს შესაბამისი ოპერაციის Id-ს.

  Response:


```
{
   "inventory_out": {
      "id": 6536,
      "date": "2019-04-23T15:07:53",
      "num_pfx": "",
      "num": 23,
      "waybill_num": null,
      "purpose": "ძირითადი საშუალების გაყიდვა № 23",
      "amount": 114.0,
      "currency": "GEL",
      "rate": 1.0,
      "store": 2,
      "customer": 8,
      "user": 1,
      "staff": 1,
      "project": 2,
      "is_vat": true,
      "make_entry": true,
      "pay_type": 2,
      "w_type": 2,
      "t_type": 1,
      "t_payer": 1,
      "w_cost": 0.0,
      "foreign": false,
      "drv_name": "",
      "tr_start": "",
      "tr_end": "empty",
      "driver_id": "",
      "car_num": "",
      "tr_text": "",
      "add_fields": [{
         "field": "usr_column_510",
         "value": ""
      }, {
         "field": "usr_column_527",
         "value": ""
      }, {
```

```
            "field": "usr_column_528",
            "value": ""
         }],
         "inventories": [{
            "id": 24,
            "price": 114.0,
            "self_cost": 5000.0,
            "quantity": 1.0
         }]
      },
      "ex": null
   }
```


სადაც:

inventory_out - ძირითადი საშუალების რეალიზაციაა რომელიც შედგება:

id (int) - დოკუმენტის id,

date (DateTime) - ოპერაციის თარიღი,

num_pfx (string[20]) - დოკუმენტის ნომრის პრეფიქსი,

num (long) - დოკუმენტის ნომერი,

waybill_num (string) - ზედნადების ნომერი (RS),

purpose (string) - დოკუმენტის შინაარსი (კომენტარი),

amount (decimal) - ოპერაციის ჯამური ღირებულება,

currency (string) - ვალუტის კოდი,

rate (decimal) - ვალუტის კურსი,

store (int) - საწყობის Id,

customer (int) - მყიდველის Id,

user (int) - მომხმარებლის (შემქმნელი) Id,

staff (int) - თანამშრომლის Id,

project (int) - პროექტის Id,

is_vat (bool) - დღგ-ს შეიცავს თუ არა,

make_entry (bool) - ბუღალტრული გატარების სტატუსი,

pay_type (byte) - გადახდის ტიპი (0 - ნაღდი, 1 - უნაღდო, 2 - კონსიგნაცია, 3 - განვადება, 4 - ნაღდი/უნაღდო, 5 - უსასყიდლო, 6 - სხვა),

w_type (byte) - ზედნადების ტიპი (2 - ტრანსპორტირებით, 3 - ტრანსპორტირების გარეშე),

t_type (byte) - ტრანსპორტირების ტიპი (1 - საავტომობილო, 2 - სარკინიგზო, 3 - საავიაციო, 4 - სხვა, 6 - საავტომობილო უცხო ქვეყნის, 7 - გადამზიდავი საავტომობილო),

t_payer (byte) - ტრანსპორტირების ღირებულების გადამხდელი (1 - მყიდველი, 2 - გამყიდველი),

w_cost (decimal) - ტრანსპორტირების ხარჯი,

foreign (bool) - უცხო ქვეყნის მოქალაქე,

drv_name (string) - მძღოლის სახელი, გვარი,

tr_start (string) - ტრანსპორტირების დაწყების ადგილი,

tr_end (string) - ტრანსპორტირების დასრულების ადგილი,

driver_id (string) - მძღოლის პირადი ნომერი,

car_num (string) - ავტომობილის ნომერი,

tr_text (string) - მისაბმელი/გადამზიდავი (თუ ტრანსპორტირების ტიპი არის 7, გადამზიდავის საიდენტიფიკაციო კოდი, თუ ტრანსპორტირების ტიპი არის 4 - ტრანსპორტირების ფორმა, ხოლო საავტომობილოს შემთხვევაში - მისაბმელის ნომერი ასეთის არსებობის შემთხვევაში),

add_fields - დამატებითი ველების კოლექცია სადაც:

field (string) - დამატებითი ველის დასახელება,

value (string) - დამატებითი ველის მნიშვნელობა,

inventories - კოლექცია შედგება:

id (int) - ძირითადი საშუალების Id,

quantity (decimal) - ძირითადი საშუალების რაოდენობა,

price (decimal) - ძირითადი საშუალების ერთეულის ფასი,

self_cost (decimal) - ძირითადი საშუალების ერთეულის თვითღირებულება.

ex (string) - ინფორმაცია შეცდომის შესახებ (ასეთის არსებობის შემთხვევაში).

getDocProductMove - საქონლის გადატანა

- **ფუნქცია:** getDocProductMove

  აღწერა: საქონლის გადატანის წამოღება

  მეთოდი: GET

  გამოძახება: api/operation/getDocProductMove/{id}

  სადაც: id (int) - წარმოადგენს შესაბამისი ოპერაციის Id-ს.

  Response:


```
{
   "product_move": {
      "id": 14830,
      "date": "2020-03-18T10:24:13",
      "num_pfx": "",
      "num": 11,
      "waybill_num": null,
      "purpose": "საქონლის გადატანა № 11",
      "amount": 18.69858,
      "store_from": 1,
      "store_to": 2,
      "user": 1,
      "staff": 2,
      "make_entry": true,
      "t_type": 1,
      "t_payer": 1,
      "w_cost": 0.0,
      "foreign": false,
      "drv_name": "",
      "tr_start": "",
      "tr_end": "",
      "driver_id": "",
      "car_num": "",
      "tr_text": "",
      "add_fields": [{
         "field": "usr_column_518",
         "value": "12:30"
      }],
      "products": [{
         "id": 12,
         "sub_id": 0,
         "self_cost": 1.01090475053712,
         "quantity": 5.0
      }, {
         "id": 13,
         "sub_id": 0,
         "self_cost": 1.94915122076992,
```

```
            "quantity": 7.0
        }]
    },
    "ex": null
}
```


სადაც:

product_move - საქონლის შიდა გადატანაა რომელიც შედგება:

id (int) - დოკუმენტის id,

date (DateTime) - ოპერაციის თარიღი,

num_pfx (string[20]) - დოკუმენტის ნომრის პრეფიქსი,

num (long) - დოკუმენტის ნომერი,

waybill_num (string) - ზედნადების ნომერი (RS),

purpose (string) - დოკუმენტის შინაარსი (კომენტარი),

amount (decimal) - ოპერაციის ჯამური ღირებულება,

store_from (int) - საწყობის Id (გამგზავნი),

store_to (int) - საწყობის Id (მიმღები),

user (int) - მომხმარებლის (შემქმნელი) Id,

staff (int) - თანამშრომლის Id,

make_entry (bool) - ბუღალტრული გატარების სტატუსი,

t_type (byte) - ტრანსპორტირების ტიპი (1 - საავტომობილო, 2 - სარკინიგზო, 3 - საავიაციო, 4 - სხვა, 6 - საავტომობილო უცხო ქვეყნის, 7 - გადამზიდავი საავტომობილო),

t_payer (byte) - ტრანსპორტირების ღირებულების გადამხდელი (1 - მყიდველი, 2 - გამყიდველი),

w_cost (decimal) - ტრანსპორტირების ხარჯი,

foreign (bool) - უცხო ქვეყნის მოქალაქე,

drv_name (string) - მძღოლის სახელი, გვარი,

tr_start (string) - ტრანსპორტირების დაწყების ადგილი,

tr_end (string) - ტრანსპორტირების დასრულების ადგილი,

driver_id (string) - მძღოლის პირადი ნომერი,

car_num (string) - ავტომობილის ნომერი,

tr_text (string) - მისაბმელი/გადამზიდავი (თუ ტრანსპორტირების ტიპი არის 7, გადამზიდავის საიდენტიფიკაციო კოდი, თუ ტრანსპორტირების ტიპი არის 4 - ტრანსპორტირების ფორმა, ხოლო საავტომობილოს შემთხვევაში - მისაბმელის ნომერი ასეთის არსებობის შემთხვევაში),

add_fields - დამატებითი ველების კოლექცია სადაც:

field (string) - დამატებითი ველის დასახელება,

value (string) - დამატებითი ველის მნიშვნელობა,

products - კოლექცია შედგება:

id (int) - საქონლის Id,

sub_id (int) - საქონლის ქვე-კოდის Id,

self_cost (decimal) - საქონლის თვითღირებულება,

quantity (decimal) - საქონლის რაოდენობა.

ex (string) - ინფორმაცია შეცდომის შესახებ (ასეთის არსებობის შემთხვევაში).

getDocInventoryMove - ძირითადი საშუალების გადატანა

- **ფუნქცია:** getDocInventoryMove

  აღწერა: ძირითადი საშუალების გადატანის წამოღება

  მეთოდი: GET

  გამოძახება: api/operation/getDocInventoryMove/{id}

  სადაც: id (int) - წარმოადგენს შესაბამისი ოპერაციის Id-ს.

  Response:


```
{
   "inventory_move": {
      "id": 14831,
      "date": "2020-03-18T10:25:25",
      "num_pfx": "",
      "num": 12,
      "waybill_num": null,
      "purpose": "ძირითადი საშუალების გადაადგილება № 12",
      "amount": 5.7,
      "store_from": 1,
      "store_to": 6,
      "user": 1,
      "staff": 2,
      "make_entry": true,
      "t_type": 1,
      "t_payer": 1,
      "w_cost": 0.0,
      "foreign": false,
      "drv_name": "",
      "tr_start": "",
      "tr_end": "",
      "driver_id": "",
      "car_num": "",
      "tr_text": "",
      "add_fields": [{
         "field": "usr_column_518",
         "value": "01"
      }],
      "inventories": [{
         "id": 34,
         "self_cost": 1.9,
         "quantity": 3.0
      }]
   },
   "ex": null
}
```

სადაც:

inventory_move - ძირ. საშუალების შიდა გადატანაა რომელიც შედგება:

id (int) - დოკუმენტის id,

date (DateTime) - ოპერაციის თარიღი,

num_pfx (string[20]) - დოკუმენტის ნომრის პრეფიქსი,

num (long) - დოკუმენტის ნომერი,

waybill_num (string) - ზედნადების ნომერი (RS),

purpose (string) - დოკუმენტის შინაარსი (კომენტარი),

amount (decimal) - ოპერაციის ჯამური ღირებულება,

store_from (int) - საწყობის Id (გამგზავნი),

store_to (int) - საწყობის Id (მიმღები),

user (int) - მომხმარებლის (შემქმნელი) Id,

staff (int) - თანამშრომლის Id,

make_entry (bool) - ბუღალტრული გატარების სტატუსი,

t_type (byte) - ტრანსპორტირების ტიპი (1 - საავტომობილო, 2 - სარკინიგზო, 3 - საავიაციო, 4 - სხვა, 6 - საავტომობილო უცხო ქვეყნის, 7 - გადამზიდავი საავტომობილო),

t_payer (byte) - ტრანსპორტირების ღირებულების გადამხდელი (1 - მყიდველი, 2 - გამყიდველი),

w_cost (decimal) - ტრანსპორტირების ხარჯი,

foreign (bool) - უცხო ქვეყნის მოქალაქე,

drv_name (string) - მძღოლის სახელი, გვარი,

tr_start (string) - ტრანსპორტირების დაწყების ადგილი,

tr_end (string) - ტრანსპორტირების დასრულების ადგილი,

driver_id (string) - მძღოლის პირადი ნომერი,

car_num (string) - ავტომობილის ნომერი,

tr_text (string) - მისაბმელი/გადამზიდავი (თუ ტრანსპორტირების ტიპი არის 7, გადამზიდავის საიდენტიფიკაციო კოდი, თუ ტრანსპორტირების ტიპი არის 4 - ტრანსპორტირების ფორმა, ხოლო საავტომობილოს შემთხვევაში - მისაბმელის ნომერი ასეთის არსებობის შემთხვევაში),

add_fields - დამატებითი ველების კოლექცია სადაც:

field (string) - დამატებითი ველის დასახელება,

value (string) - დამატებითი ველის მნიშვნელობა,

inventories - კოლექცია შედგება:

id (int) - ძირ. საშუალების Id,

self_cost (decimal) - ძირ. საშუალების თვითღირებულება,

quantity (decimal) - ძირ. საშუალების რაოდენობა.

ex (string) - ინფორმაცია შეცდომის შესახებ (ასეთის არსებობის შემთხვევაში).

getDocCustomerInventoryReturn - ძირითადი საშუალების დაბრუნება მყიდველისგან

- **ფუნქცია:** getDocCustomerInventoryReturn

  აღწერა: მყიდველისგან მობრუნებული ძირითადი საშუალების წამოღება

  მეთოდი: GET

  გამოძახება: api/operation/getDocCustomerInventoryReturn/{id}

  სადაც: id (int) - წარმოადგენს შესაბამისი ოპერაციის Id-ს.

  Response:


```
{
   "customer_inventory_return": {
      "id": 8743,
      "date": "2019-11-12T11:24:37",
      "num_pfx": "",
      "num": 11,
      "waybill_num": null,
      "purpose": "ძირითადი საშუალების უკან დაბრუნება № 0",
      "amount": 20.0,
      "currency": "GEL",
      "rate": 1.0,
      "store": 2,
      "customer": 30,
      "user": 1,
      "staff": 1,
      "project": 2,
      "is_vat": true,
      "make_entry": true,
      "pay_type": 1,
      "t_type": 1,
      "t_payer": 1,
      "w_cost": 0.0,
      "foreign": false,
      "drv_name": "",
      "tr_start": "",
      "tr_end": "",
      "driver_id": "",
      "car_num": "",
      "tr_text": "",
      "inventories": [{
         "id": 34,
         "price": 3.5,
         "self_cost": 1.9,
         "quantity": 2.0
      }]
   },
```

```
   "ex": null
}
```


სადაც:

customer_inventory_return - მყიდველისგან ძირ. საშ. დაბრუნებაა რომელიც შედგება:

id (int) - დოკუმენტის id,

date (DateTime) - ოპერაციის თარიღი,

num_pfx (string[20]) - დოკუმენტის ნომრის პრეფიქსი,

num (long) - დოკუმენტის ნომერი,

waybill_num (string) - ზედნადების ნომერი (RS),

purpose (string) - დოკუმენტის შინაარსი (კომენტარი),

amount (decimal) - ოპერაციის ჯამური ღირებულება,

currency (string) - ვალუტის კოდი,

rate (decimal) - ვალუტის კურსი,

store (int) - საწყობის Id,

customer (int) - მყიდველის Id,

user (int) - მომხმარებლის (შემქმნელი) Id,

staff (int) - თანამშრომლის Id,

project (int) - პროექტის Id,

is_vat (bool) - დღგ-ს შეიცავს თუ არა,

make_entry (bool) - ბუღალტრული გატარების სტატუსი,

pay_type (byte) - გადახდის ტიპი (0 - ნაღდი, 1 - უნაღდო, 2 - კონსიგნაცია, 3 - განვადება, 4 - ნაღდი/უნაღდო, 5 - უსასყიდლო, 6 - სხვა),

t_type (byte) - ტრანსპორტირების ტიპი (1 - საავტომობილო, 2 - სარკინიგზო, 3 - საავიაციო, 4 - სხვა, 6 - საავტომობილო უცხო ქვეყნის, 7 - გადამზიდავი საავტომობილო),

t_payer (byte) - ტრანსპორტირების ღირებულების გადამხდელი (1 - მყიდველი, 2 - გამყიდველი),

w_cost (decimal) - ტრანსპორტირების ხარჯი,

foreign (bool) - უცხო ქვეყნის მოქალაქე,

drv_name (string) - მძღოლის სახელი, გვარი,

tr_start (string) - ტრანსპორტირების დაწყების ადგილი,

tr_end (string) - ტრანსპორტირების დასრულების ადგილი,

driver_id (string) - მძღოლის პირადი ნომერი,

car_num (string) - ავტომობილის ნომერი,

tr_text (string) - მისაბმელი/გადამზიდავი (თუ ტრანსპორტირების ტიპი არის 7, გადამზიდავის საიდენტიფიკაციო კოდი, თუ ტრანსპორტირების ტიპი არის 4 - ტრანსპორტირების

ფორმა, ხოლო საავტომობილოს შემთხვევაში - მისაბმელის ნომერი ასეთის არსებობის შემთხვევაში),

inventories - კოლექცია შედგება:

id (int) - ძირითადი საშუალების Id,

quantity (decimal) - ძირითადი საშუალების რაოდენობა,

price (decimal) - ძირითადი საშუალების ერთეულის ფასი,

self_cost (decimal) - ძირითადი საშუალების ერთეულის (დაბრუნების) თვითღირებულება.

ex (string) - ინფორმაცია შეცდომის შესახებ (ასეთის არსებობის შემთხვევაში).

getDocProvidedService - მომსახურების გაწევა

- **ფუნქცია:** getDocProvidedService

  აღწერა: გაწეული მომსახურების წამოღება

  მეთოდი: GET

  გამოძახება: api/operation/getDocProvidedService /{id}

  სადაც: id (int) - წარმოადგენს შესაბამისი ოპერაციის Id-ს.

  Response:


```
{
   "doc_providedservice": {
      "id": 6564,
      "date": "2019-05-29T16:10:26",
      "num_pfx": "",
      "num": 9,
      "purpose": "მომსახურების გაწევა № 9",
      "amount": 240.0,
      "currency": "GEL",
      "rate": 1.0,
      "store": 1,
      "customer": 6,
      "user": 1,
      "staff": 1,
      "project": 1,
      "is_vat": true,
      "make_entry": true,
      "pay_type": 1,
      "overlap_type": 3,
      "overlap_amount": 0.0,
      "add_fields": [{
         "field": "usr_column_510",
         "value": ""
      }, {
         "field": "usr_column_528",
         "value": ""
      }],
      "services": [{
         "id": 11,
         "price": 120.0,
         "quantity": 2.0
      }]
   },
   "ex": null
}
```

სადაც:

doc_providedservice - მომსახურების გაწევის ოპერაციაა რომელიც შედგება:

id (int) - დოკუმენტის id,

date (DateTime) - ოპერაციის თარიღი,

num_pfx (string[20]) - დოკუმენტის ნომრის პრეფიქსი,

num (long) - დოკუმენტის ნომერი,

purpose (string) - დოკუმენტის შინაარსი (კომენტარი),

amount (decimal) - ოპერაციის ჯამური ღირებულება,

currency (string) - ვალუტის კოდი,

rate (decimal) - ვალუტის კურსი,

store (int) - საწყობის Id,

customer (int) - მყიდველის Id,

user (int) - მომხმარებლის (შემქმნელი) Id,

staff (int) - თანამშრომლის Id,

project (int) - პროექტის Id,

is_vat (bool) - დღგ-ს შეიცავს თუ არა,

make_entry (bool) - ბუღალტრული გატარების სტატუსი,

pay_type (byte) - გადახდის ტიპი (0 - ნაღდი, 1 - უნაღდო, 2 - კონსიგნაცია, 3 - განვადება, 4 - ნაღდი/უნაღდო, 5 - უსასყიდლო, 6 - სხვა),

overlap_type (byte) - ავანსი/დღგ-ს გადახურვა (0 - არ გადაიხუროს არც ერთი, 1 - ავანსი გადაიხუროს, დღგ - ნაწილობრივ, 2 - ავანსი გადაიხუროს, დღგ - სრულად, 3 - ავანსი გადაიხუროს, დღგ - არა) ,

overlap_amount (decimal) - გადახურული დღგ- ს თანხა ,

add_fields - დამატებითი ველების კოლექცია სადაც:

field (string) - დამატებითი ველის დასახელება,

value (string) - დამატებითი ველის მნიშვნელობა,

services - კოლექცია შედგება:

id (int) - გაწეული მომსახურების Id,

quantity (decimal) - გაწეული მომსახურების რაოდენობა,

price (decimal) - გაწეული მომსახურების ერთეულის ფასი.

ex (string) - ინფორმაცია შეცდომის შესახებ (ასეთის არსებობის შემთხვევაში).

getDocReceivedService - მომსახურების მიღება

- **ფუნქცია:** getDocReceivedService

  აღწერა: მიღებული მომსახურების წამოღება

  მეთოდი: GET

  გამოძახება: api/operation/getDocReceivedService/{id}

  სადაც: id (int) - წარმოადგენს შესაბამისი ოპერაციის Id-ს.

  Response:


```
{
   "doc_receivedservice": {
      "id": 10756,
      "date": "2019-12-09T16:22:04",
      "num_pfx": "",
      "num": 5,
      "purpose": "მომსახურების მიღება № 5",
      "amount": 2000.0,
      "currency": "GEL",
      "rate": 1.0,
      "vendor": 24,
      "user": 1,
      "staff": 0,
      "project": 1,
      "is_vat": false,
      "make_entry": true,
      "pay_type": 2,
      "overlap_type": 2,
      "overlap_amount": 0.0,
      "add_fields": [{
         "field": "usr_column_525",
         "value": "uu"
      }, {
         "field": "usr_column_536",
         "value": "0.12.2"
      }],
      "services": [{
         "id": 7,
         "price": 1000.000,
         "quantity": 2.000
      }]
   },
   "ex": null
}
```

სადაც:

doc_receivedservice - მომსახურების მიღების ოპერაციაა რომელიც შედგება:

id (int) - დოკუმენტის id,

date (DateTime) - ოპერაციის თარიღი,

num_pfx (string[20]) - დოკუმენტის ნომრის პრეფიქსი,

num (long) - დოკუმენტის ნომერი,

purpose (string) - დოკუმენტის შინაარსი (კომენტარი),

amount (decimal) - ოპერაციის ჯამური ღირებულება,

currency (string) - ვალუტის კოდი,

rate (decimal) - ვალუტის კურსი,

vendor (int) - მომწოდებელი Id,

user (int) - მომხმარებლის (შემქმნელი) Id,

staff (int) - თანამშრომლის Id,

project (int) - პროექტის Id,

is_vat (bool) - დღგ-ს შეიცავს თუ არა,

make_entry (bool) - ბუღალტრული გატარების სტატუსი,

pay_type (byte) - გადახდის ტიპი (0 - ნაღდი, 1 - უნაღდო, 2 - კონსიგნაცია, 3 - განვადება, 4 - ნაღდი/უნაღდო, 5 - უსასყიდლო, 6 - სხვა),

overlap_type (byte) - ავანსი/დღგ-ს გადახურვა (0 - არ გადაიხუროს არც ერთი, 1 - ავანსი გადაიხუროს, დღგ - ნაწილობრივ, 2 - ავანსი გადაიხუროს, დღგ - სრულად, 3 - ავანსი გადაიხუროს, დღგ - არა) ,

overlap_amount (decimal) - გადახურული დღგ- ს თანხა ,

add_fields - დამატებითი ველების კოლექცია სადაც:

field (string) - დამატებითი ველის დასახელება,

value (string) - დამატებითი ველის მნიშვნელობა,

services - კოლექცია შედგება:

id (int) - მიღებული მომსახურების Id,

quantity (decimal) - მიღებული მომსახურების რაოდენობა,

price (decimal) - მიღებული მომსახურების ერთეულის ფასი.

ex (string) - ინფორმაცია შეცდომის შესახებ (ასეთის არსებობის შემთხვევაში).

getDocCustomerReturn - დაბრუნება მყიდველისგან

- **ფუნქცია:** getDocCustomerReturn

  აღწერა: მყიდველისგან მობრუნებული საქონლის წამოღება

  მეთოდი: GET

  გამოძახება: api/operation/getDocCustomerReturn/{id}

  სადაც: id (int) - წარმოადგენს შესაბამისი ოპერაციის Id-ს.

  Response:


```
{
   "customer_return": {
      "id": 6552,
      "date": "2019-05-01T14:25:00",
      "num_pfx": "",
      "num": 10,
      "waybill_num": "0526302888",
      "purpose": "dabruneba",
      "amount": 65.7,
      "currency": "GEL",
      "rate": 1.0,
      "store": 1,
      "customer": 8,
      "user": 2,
      "staff": 3,
      "project": 2,
      "is_vat": true,
      "make_entry": true,
      "pay_type": 1,
      "t_type": 1,
      "t_payer": 2,
      "w_cost": 0.5,
      "foreign": false,
      "drv_name": "მძღოლის სახელი, გვარი",
      "tr_start": "ტრანსპორტირების დაწყების ადგილი",
      "tr_end": "ტრანსპორტირების დასრულების ადგილი",
      "driver_id": "12345678910",
      "car_num": "SAK001",
      "tr_text": "მისაბმელი/გადამზიდავი",
      "products": [{
         "id": 29,
         "sub_id": 0,
         "price": 35.7,
         "self_cost": 1.69491525423729,
         "quantity": 1.0,
         "out_id": 6547
      }, {
```

```
            "id": 3,
            "sub_id": 0,
            "price": 10.0,
            "self_cost": 8.47457627118644,
            "quantity": 3.0,
            "out_id": 0
        }]
    },
    "ex": null
}
```


სადაც:

customer_return - მყიდველისგან დაბრუნებაა რომელიც შედგება:

id (int) - დოკუმენტის id,

date (DateTime) - ოპერაციის თარიღი,

num_pfx (string[20]) - დოკუმენტის ნომრის პრეფიქსი,

num (long) - დოკუმენტის ნომერი,

waybill_num (string) - ზედნადების ნომერი (RS),

purpose (string) - დოკუმენტის შინაარსი (კომენტარი),

amount (decimal) - ოპერაციის ჯამური ღირებულება,

currency (string) - ვალუტის კოდი,

rate (decimal) - ვალუტის კურსი,

store (int) - საწყობის Id,

customer (int) - მყიდველის Id,

user (int) - მომხმარებლის (შემქმნელი) Id,

staff (int) - თანამშრომლის Id,

project (int) - პროექტის Id,

is_vat (bool) - დღგ-ს შეიცავს თუ არა,

make_entry (bool) - ბუღალტრული გატარების სტატუსი,

pay_type (byte) - გადახდის ტიპი (0 - ნაღდი, 1 - უნაღდო, 2 - კონსიგნაცია, 3 - განვადება, 4 - ნაღდი/უნაღდო, 5 - უსასყიდლო, 6 - სხვა),

t_type (byte) - ტრანსპორტირების ტიპი (1 - საავტომობილო, 2 - სარკინიგზო, 3 - საავიაციო, 4 - სხვა, 6 - საავტომობილო უცხო ქვეყნის, 7 - გადამზიდავი საავტომობილო),

t_payer (byte) - ტრანსპორტირების ღირებულების გადამხდელი (1 - მყიდველი, 2 - გამყიდველი),

w_cost (decimal) - ტრანსპორტირების ხარჯი,

foreign (bool) - უცხო ქვეყნის მოქალაქე,

drv_name (string) - მძღოლის სახელი, გვარი,

tr_start (string) - ტრანსპორტირების დაწყების ადგილი,

tr_end (string) - ტრანსპორტირების დასრულების ადგილი,

driver_id (string) - მძღოლის პირადი ნომერი,

car_num (string) - ავტომობილის ნომერი,

tr_text (string) - მისაბმელი/გადამზიდავი (თუ ტრანსპორტირების ტიპი არის 7, გადამზიდავის საიდენტიფიკაციო კოდი, თუ ტრანსპორტირების ტიპი არის 4 - ტრანსპორტირების ფორმა, ხოლო საავტომობილოს შემთხვევაში - მისაბმელის ნომერი ასეთის არსებობის შემთხვევაში),

products - კოლექცია შედგება:

id (int) - საქონლის Id,

sub_id (int) - საქონლის ქვე-კოდის Id,

price (decimal) - საქონლის ერთეულის ფასი,

self_cost (decimal) - საქონლის ერთეულის (დაბრუნების) თვითღირებულება,

quantity (decimal) - საქონლის რაოდენობა,

out_id (int) - გაყიდვის ოპერაციის Id (თუ რომელი გაყიდვის დაბრუნება ხდება კონკრეტულად, default=0).

ex (string) - ინფორმაცია შეცდომის შესახებ (ასეთის არსებობის შემთხვევაში).

updateRsStatus - არსებული ზედნადების RS სტატუსის განახლება

- **ფუნქცია:** updateRsStatus

  აღწერა: არსებული ზედნადების RS სტატუსის განახლება

  მეთოდი: POST

  გამოძახება: api/operation/updateRsStatus

Request Body:

```
{
   "id": 4164,
   "w_id": 352264608,
   "w_status": 1,
   "w_num": "0123654878"
}
```

სადაც:

id (int) - FINA ს ბაზაში არსებული ოპერაციის Id,

w_id (long) - ზედნადების Id, რომელიც მიენიჭა RS.ge ზე ატვირთვისას,

w_status (int) - ზედნადების სტატუსი, რომელიც მიენიჭა RS.ge ზე,

w_num (string[50]) - ზედნადების ნომერი, რომელიც მიენიჭა RS.ge ზე ატვირთვისას.

Response:

```
{
   "status": true,
   "ex": null
}
```

სადაც:

status (bool) აბრუნებს მნიშვნელობას ოპერაციის წარმატებით შესრულების მიხედვით,

ex (string) - ინფორმაცია შეცდომის შესახებ (ასეთის არსებობის შემთხვევაში).

saveDocDiscountCard - ფასდაკლების ბარათის გაცემა

- **ფუნქცია:** saveDocDiscountCard

  აღწერა: მყიდველზე გაცემული ფასდაკლების ბარათის შენახვა (insert, update)

  მეთოდი: POST

  გამოძახება: api/operation/saveDocDiscountCard

  Request Body:

```
{
   "id": 0,
   "date": "2019-11-08T18:00:00",
   "num_pfx": "",
   "doc_num": 0,
   "purpose": "ფასდაკლების ბარათის გაცემა",
   "customer": 31,
   "store": 1,
   "user": 1,
   "card_code": "231",
   "discount_id": 1,
   "status": true
}
```

  სადაც:

  id (int) - ოპერაციის Id. (თუ იქმნება ახალი, გადაეცემა 0),

  date (DateTime) - ოპერაციის თარიღი,

  num_pfx (string[20]) - დოკუმენტის ნომრის პრეფიქსი,

  num (long) - დოკუმენტის ნომერი,

  purpose (string[750]) - დოკუმენტის შინაარსი (კომენტარი),

  customer (int) - მყიდველის Id,

  store (int) - საწყობის Id,

  user (int) - მომხმარებლის (შემქმნელი) Id,

  code (string) - ფასდაკლების ბარათის კოდი,

  discount (int) - ფასდაკლების Id (იხ მეთოდი getDiscountTypes),

  status (bool) - ბარათის სტატუსი (აქტიურია თუ არა)

  Response:

```
{
   "id": 2,
   "ex": null
}
```

სადაც:

id (int) - დამატებული (ან დარედაქტირებული) ოპერაციის Id,

ex (string) - ინფორმაცია შეცდომის შესახებ (ასეთის არსებობის შემთხვევაში).

saveDocGiftCard - სასაჩუქრე ბარათის გაცემა

- **ფუნქცია:** saveDocGiftCard

  აღწერა: გაცემული სასაჩუქრე ბარათის შენახვა (insert)

  მეთოდი: POST

  გამოძახება: api/operation/saveDocGiftCard

  Request Body:

```
{
   "date": "2019-11-11T18:00:00",
   "amount": 100,
   "pay_amount": 100,
   "code": "TT3359",
   "store": 1,
   "user": 1
}
```

  სადაც:

  date (DateTime) - ოპერაციის თარიღი,

  amount (decimal) - სასაჩუქრე ბარათის ღირებულება,

  pay_amount (decimal) - რეალურად გადახდილი თანხის ოდენობა,

  code (string) - ბარათის კოდი,

  store (int) - საწყობის (მაღაზიის) Id,

  user (int) - მომხმარებლის (შემქმნელი) Id,

  Response:

```
{
   "id": 2,
   "ex": null
}
```

  სადაც:

  id (int) - დამატებული სასაჩუქრე ბარათის Id,

  ex (string) - ინფორმაცია შეცდომის შესახებ (ასეთის არსებობის შემთხვევაში).

saveDocBonusOperation - ქულის დაგროვება/გახარჯვა

- **ფუნქცია:** saveDocBonusOperation

  აღწერა: ქულის დაგროვების ან გახარჯვის შენახვა (insert, update)

  მეთოდი: POST

  გამოძახება: api/operation/saveDocBonusOperation

  Request Body:

```
{
   "card_id": 2013,
   "ref_id": 16975,
   "coeff": 1,
   "amount": 10.0
}
```

  სადაც:

  card_id (int) - დაგროვების ბარათის Id,

  ref_id (int) - ოპერაციის Id, რომლის საფუძველზეც ხდება შესაბამისი ქულის დაგროვება ან გახარჯვა,

  coeff (sbyte) - კოეფიციენტი 1 ან -1 (იმისდა მიხედვით დაგროვება ხდება თუ გახარჯვა),

  amount (decimal) - დაგროვილი (გახარჯული) ქულის ოდენობა (გამოთვლილი თანხაში)

  Response:

```
{
   "res": true,
   "ex": null
}
```

  სადაც:

  res (bool) - ოპერაციის შესრულების შედეგი,

  ex (string) - ინფორმაცია შეცდომის შესახებ (ასეთის არსებობის შემთხვევაში).

saveDocCustomerOrder - მყიდველისგან მიღებული შეკვეთა

- **ფუნქცია:** saveDocCustomerOrder

  აღწერა: მყიდველისგან მიღებული შეკვეთის შენახვა (insert, update)

  მეთოდი: POST

  გამოძახება: api/operation/saveDocCustomerOrder

  Request Body:

```
{
   "id": 0,
   "date": "2018-12-04T18:00:00",
   "num_pfx": "",
   "doc_num": 0,
   "purpose": "შეკვეთა მყიდველისგან - Test",
   "amount": 65.7,
   "currency": "GEL",
   "rate": 1.0,
   "store": 1,
   "user": 2,
   "staff": 3,
   "project": 2,
   "customer": 8,
   "is_vat": true,
   "pay_type": 1,
   "tr_start": "ტრანსპორტირების დაწყების ადგილი",
   "tr_end": "ტრანსპორტირების დასრულების ადგილი",
   "reserved": true,
   "actived": true,
   "add_fields": [{
      "field": "usr_column_511",
      "value": "ირმის ნახტომი"
   }],
   "products": [{
      "id": 2,
      "sub_id": 1,
      "quantity": 1.0,
      "price": 35.70
   }, {
      "id": 3,
      "sub_id": 2,
      "quantity": 3.0,
      "price": 10.0
   }],
   "services": []
}
```

სადაც:

id (int) - ოპერაციის Id. (თუ იქმნება ახალი, გადაეცემა 0),

date (DateTime) - ოპერაციის თარიღი,

num_pfx (string[20]) - დოკუმენტის ნომრის პრეფიქსი,

num (long) - დოკუმენტის ნომერი,

purpose (string[750]) - დოკუმენტის შინაარსი (კომენტარი),

amount (decimal) - ოპერაციის ჯამური ღირებულება,

currency (string) - ვალუტის კოდი,

rate (decimal) - ვალუტის კურსი,

store (int) - საწყობის Id,

user (int) - მომხმარებლის (შემქმნელი) Id,

staff (int) - თანამშრომლის Id (0 = მომხმარებელზე მიბმული თანამშრომლის აღება),

project (int) - პროექტის Id,

customer (int) - მყიდველის Id,

is_vat (bool) - დღგ-ს შეიცავს თუ არა,

pay_type (byte) - გადახდის ტიპი (0 - ნაღდი, 1 - უნაღდო, 2 - კონსიგნაცია, 3 - განვადება, 4 - ნაღდი/უნაღდო, 5 - უსასყიდლო, 6 - სხვა),

tr_start (string[200]) - ტრანსპორტირების დაწყების ადგილი,

tr_end (string[200]) - ტრანსპორტირების დასრულების ადგილი,

reserved (bool) - რეზერვაციის სტატუსი.

actived (bool) - შეკვეთის სტატუსი (აქტიურია თუ არა).

add_fields - დამატებითი ველების კოლექცია სადაც:

field (string) - დამატებითი ველის დასახელება,

value (string) - დამატებითი ველის მნიშვნელობა,

products - კოლექცია შედგება:

id (int) - საქონლის Id,

sub_id (int) - საქონლის ქვე-კოდის Id, (default=0),

quantity (decimal) - საქონლის რაოდენობა,

price (decimal) - საქონლის ერთეულის ფასი,

services - კოლექცია შედგება:

id (int) - გაწეული მომსახურების Id,

quantity (decimal) - გაწეული მომსახურების რაოდენობა,

price (decimal) - გაწეული მომსახურების ერთეულის ფასი.

Response:


```
{
   "id": 2,
   "ex": null
}
```


სადაც:

id (int) - დამატებული (ან დარედაქტირებული) ოპერაციის Id,

ex (string) - ინფორმაცია შეცდომის შესახებ (ასეთის არსებობის შემთხვევაში).

- **ფუნქცია:** saveDocProductOut

  აღწერა: საქონლის რეალიზაციის შენახვა (insert, update)

  მეთოდი: POST

  გამოძახება: api/operation/saveDocProductOut

  Request Body:


```
{
   "id": 0,
   "date": "2018-12-08T18:00:00",
   "num_pfx": "",
   "doc_num": 0,
   "purpose": "რელიზაცია",
   "amount": 65.7,
   "currency": "GEL",
   "rate": 1.0,
   "store": 1,
   "user": 2,
   "staff": 3,
   "project": 2,
   "customer": 8,
   "is_vat": true,
   "make_entry": true,
   "pay_type": 1,
   "w_type": 2,
   "t_type": 1,
   "t_payer": 2,
   "w_cost": 0.5,
   "foreign": false,
   "drv_name": "მძღოლის სახელი, გვარი",
   "tr_start": "ტრანსპორტირების დაწყების ადგილი",
   "tr_end": "ტრანსპორტირების დასრულების ადგილი",
   "driver_id": "12345678910",
   "car_num": "SAK005",
   "tr_text": "მისაბმელი/გადამზიდავი",
   "overlap_type": 0,
   "overlap_amount": 0,
   "add_fields": [{
      "field": "usr_column_510",
      "value": "test string"
   }],
   "products": [{
      "id": 2,
      "sub_id": 0,
      "quantity": 1.0,
      "price": 35.70
```

```
    }, {
        "id": 3,
        "sub_id": 0,
        "quantity": 3.0,
        "price": 10.0
    }],
    "services": [{
        "id": 20,
        "quantity": 1.0,
        "price": 2.0
    }]
}
```


სადაც:

id (int) - ოპერაციის Id. (თუ იქმნება ახალი, გადაეცემა 0),

date (DateTime) - ოპერაციის თარიღი,

num_pfx (string[20]) - დოკუმენტის ნომრის პრეფიქსი,

num (long) - დოკუმენტის ნომერი,

purpose (string[750]) - დოკუმენტის შინაარსი (კომენტარი),

amount (decimal) - ოპერაციის ჯამური ღირებულება,

currency (string) - ვალუტის კოდი,

rate (decimal) - ვალუტის კურსი,

store (int) - საწყობის Id,

user (int) - მომხმარებლის (შემქმნელი) Id,

staff (int) - თანამშრომლის Id (0 = მომხმარებელზე მიბმული თანამშრომლის აღება),

project (int) - პროექტის Id,

customer (int) - მყიდველის Id,

is_vat (bool) - დღგ-ს შეიცავს თუ არა,

make_entry (bool) - შესრულდეს თუ არა ბუღალტრული გატარება,

pay_type (byte) - გადახდის ტიპი (0 - ნაღდი, 1 - უნაღდო, 2 - კონსიგნაცია, 3 - განვადება, 4 - ნაღდი/უნაღდო, 5 - უსასყიდლო, 6 - სხვა),

w_type (byte) - ზედნადების ტიპი (2 - ტრანსპორტირებით, 3 - ტრანსპორტირების გარეშე),

t_type (byte) - ტრანსპორტირების ტიპი (1 - საავტომობილო, 2 - სარკინიგზო, 3 - საავიაციო, 4 - სხვა, 6 - საავტომობილო უცხო ქვეყნის, 7 - გადამზიდავი საავტომობილო),

t_payer (byte) - ტრანსპორტირების ღირებულების გადამხდელი (1 - მყიდველი, 2 - გამყიდველი),

w_cost (decimal) - ტრანსპორტირების ხარჯი,

foreign (bool) - უცხო ქვეყნის მოქალაქე,

drv_name (string[50]) - მძღოლის სახელი, გვარი,

tr_start (string[200]) - ტრანსპორტირების დაწყების ადგილი,

tr_end (string[200]) - ტრანსპორტირების დასრულების ადგილი,

driver_id (string[50]) - მძღოლის პირადი ნომერი,

car_num (string[50]) - ავტომობილის ნომერი,

tr_text (string[200]) - მისაბმელი/გადამზიდავი (თუ ტრანსპორტირების ტიპი არის 7, უნდა მიეთითოს გადამზიდავის საიდენტიფიკაციო კოდი, თუ ტრანსპორტირების ტიპი არის 4 - ეთიება ტრანსპორტირების ფორმა, ხოლო საავტომობილოს შემთხვევაში - მისაბმელის ნომერი ასეთის არსებობის შემთხვევაში),

overlap_type (byte) - ავანსი/დღგ-ს გადახურვა (0 - არ გადაიხუროს არც ერთი, 1 - ავანსი გადაიხუროს, დღგ - ნაწილობრივ, 2 - ავანსი გადაიხუროს, დღგ - სრულად, 3 - ავანსი გადაიხუროს, დღგ - არა) ,

overlap_amount (decimal) - გადახურული დღგ- ს თანხა, (თუ overlap_type არის 2, მოხდება თანხის ავტომატური დაანგარიშება, თუ overlap_type არის 1 - გადახურული დღგ -ს თანხა არ უნდა აღემატებოდეს მყიდველის ავანსის დღგ-ს თანხას),

add_fields - დამატებითი ველების კოლექცია სადაც:

field (string) - დამატებითი ველის დასახელება,

value (string) - დამატებითი ველის მნიშვნელობა,

products - კოლექცია შედგება:

id (int) - საქონლის Id,

sub_id (int) - საქონლის ქვე-კოდის Id, (default=0),

quantity (decimal) - საქონლის რაოდენობა,

price (decimal) - საქონლის ერთეულის ფასი,

services - კოლექცია შედგება:

id (int) - გაწეული მომსახურების Id,

quantity (decimal) - გაწეული მომსახურების რაოდენობა,

price (decimal) - გაწეული მომსახურების ერთეულის ფასი.

Response:


```
{
   "id": 3,
   "ex": null
}
```


სადაც:

id (int) - დამატებული (ან დარედაქტირებული) ოპერაციის Id,

ex (string) - ინფორმაცია შეცდომის შესახებ (ასეთის არსებობის შემთხვევაში).

- **ფუნქცია:** saveDocProductMove

  აღწერა: საქონლის შიდა გადაზიდვის შენახვა (insert, update)

  მეთოდი: POST

  გამოძახება: api/operation/saveDocProductMove

  Request Body:

```
{
   "id": 0,
   "date": "2019-02-14T11:00:00",
   "num_pfx": "",
   "doc_num": 0,
   "purpose": "შიდა გადაზიდვა - Test",
   "store_from": 1,
   "store_to": 2,
   "user": 2,
   "staff": 3,
   "make_entry": true,
   "t_type": 1,
   "t_payer": 2,
   "w_cost": 0.5,
   "foreign": false,
   "drv_name": "მძღოლის სახელი, გვარი",
   "tr_start": "ტრანსპორტირების დაწყების ადგილი",
   "tr_end": "ტრანსპორტირების დასრულების ადგილი",
   "driver_id": "12345678910",
   "car_num": "SAK005",
   "tr_text": "მისაბმელი/გადამზიდავი",
   "add_fields": [{
      "field": "usr_column_518",
      "value": "test string123"
   }],
   "products": [{
      "id": 2,
      "sub_id": 0,
      "quantity": 1
   }, {
      "id": 3,
      "sub_id": 0,
      "quantity": 3
   }]
}
```

  სადაც:

  id (int) - ოპერაციის Id. (თუ იქმნება ახალი, გადაეცემა 0),

date (DateTime) - ოპერაციის თარიღი,

num_pfx (string[20]) - დოკუმენტის ნომრის პრეფიქსი,

num (long) - დოკუმენტის ნომერი,

purpose (string[750]) - დოკუმენტის შინაარსი (კომენტარი),

store_from (int) - გამგზავნი საწყობის Id,

store_to (int) - მიმღები საწყობის Id,

user (int) - მომხმარებლის (შემქმნელი) Id,

staff (int) - თანამშრომლის Id (0 = მომხმარებელზე მიბმული თანამშრომლის აღება),

make_entry (bool) - შესრულდეს თუ არა ბუღალტრული გატარება,

t_type (byte) - ტრანსპორტირების ტიპი (1 - საავტომობილო, 2 - სარკინიგზო, 3 - საავიაციო, 4 - სხვა, 6 - საავტომობილო უცხო ქვეყნის, 7 - გადამზიდავი საავტომობილო),

t_payer (byte) - ტრანსპორტირების ღირებულების გადამხდელი (1 - მყიდველი, 2 - გამყიდველი),

w_cost (decimal) - ტრანსპორტირების ხარჯი,

foreign (bool) - უცხო ქვეყნის მოქალაქე,

drv_name (string[50]) - მძღოლის სახელი, გვარი,

tr_start (string[200]) - ტრანსპორტირების დაწყების ადგილი,

tr_end (string[200]) - ტრანსპორტირების დასრულების ადგილი,

driver_id (string[50]) - მძღოლის პირადი ნომერი,

car_num (string[50]) - ავტომობილის ნომერი,

tr_text (string[200]) - მისაბმელი/გადამზიდავი (თუ ტრანსპორტირების ტიპი არის 7, უნდა მიეთითოს გადამზიდავის საიდენტიფიკაციო კოდი, თუ ტრანსპორტირების ტიპი არის 4 - ეთიება ტრანსპორტირების ფორმა, ხოლო საავტომობილოს შემთხვევაში - მისაბმელის ნომერი ასეთის არსებობის შემთხვევაში),

add_fields - დამატებითი ველების კოლექცია სადაც:

field (string) - დამატებითი ველის დასახელება,

value (string) - დამატებითი ველის მნიშვნელობა,

products - კოლექცია შედგება:

id (int) - საქონლის Id,

sub_id (int) - საქონლის ქვე-კოდის Id, (default=0),

quantity (decimal) - საქონლის რაოდენობა.

Response:


```
{
   "id": 3,
   "ex": null
}
```

სადაც:

id (int) - დამატებული (ან დარედაქტირებული) ოპერაციის Id,

ex (string) - ინფორმაცია შეცდომის შესახებ (ასეთის არსებობის შემთხვევაში).

saveDocProvidedService - მომსახურების გაწევა

- **ფუნქცია:** saveDocProvidedService

  აღწერა: გაწეული მომსახურების შენახვა (insert, update)

  მეთოდი: POST

  გამოძახება: api/operation/saveDocProvidedService

  Request Body:

```
{
   "id": 0,
   "date": "2018-12-08T18:00:00",
   "num_pfx": "",
   "doc_num": 0,
   "purpose": "მომსახურების გაწევა",
   "amount": 2.0,
   "currency": "GEL",
   "rate": 1.0,
   "store": 1,
   "user": 2,
   "staff": 3,
   "project": 2,
   "customer": 8,
   "is_vat": true,
   "make_entry": true,
   "pay_type": 1,
   "overlap_type": 0,
   "overlap_amount": 0,
   "add_fields": [{
      "field": "usr_column_510",
      "value": "test string"
   }],
   "services": [{
      "id": 20,
      "quantity": 1.0,
      "price": 2.0
   }]
}
```

  სადაც:

  id (int) - ოპერაციის Id. (თუ იქმნება ახალი, გადაეცემა 0),

  date (DateTime) - ოპერაციის თარიღი,

  num_pfx (string[20]) - დოკუმენტის ნომრის პრეფიქსი,

  num (long) - დოკუმენტის ნომერი,

  purpose (string[750]) - დოკუმენტის შინაარსი (კომენტარი),

amount (decimal) - ოპერაციის ჯამური ღირებულება,

currency (string) - ვალუტის კოდი,

rate (decimal) - ვალუტის კურსი,

store (int) - საწყობის Id,

user (int) - მომხმარებლის (შემქმნელი) Id,

staff (int) - თანამშრომლის Id (0 = მომხმარებელზე მიბმული თანამშრომლის აღება),

project (int) - პროექტის Id,

customer (int) - მყიდველის Id,

is_vat (bool) - დღგ-ს შეიცავს თუ არა,

make_entry (bool) - შესრულდეს თუ არა ბუღალტრული გატარება,

pay_type (byte) - გადახდის ტიპი (0 - ნაღდი, 1 - უნაღდო, 2 - კონსიგნაცია, 3 - განვადება, 4 - ნაღდი/უნაღდო, 5 - უსასყიდლო, 6 - სხვა),

overlap_type (byte) - ავანსი/დღგ-ს გადახურვა (0 - არ გადაიხუროს არც ერთი, 1 - ავანსი გადაიხუროს, დღგ - ნაწილობრივ, 2 - ავანსი გადაიხუროს, დღგ - სრულად, 3 - ავანსი გადაიხუროს, დღგ - არა) ,

overlap_amount (decimal) - გადახურული დღგ- ს თანხა, (თუ overlap_type არის 2, მოხდება თანხის ავტომატური დაანგარიშება, თუ overlap_type არის 1 - გადახურული დღგ -ს თანხა არ უნდა აღემატებოდეს მყიდველის ავანსის დღგ-ს თანხას),

add_fields - დამატებითი ველების კოლექცია სადაც:

field (string) - დამატებითი ველის დასახელება,

value (string) - დამატებითი ველის მნიშვნელობა,

services - კოლექცია შედგება:

id (int) - გაწეული მომსახურების Id,

quantity (decimal) - გაწეული მომსახურების რაოდენობა,

price (decimal) - გაწეული მომსახურების ერთეულის ფასი.

Response:


```
{
   "id": 3,
   "ex": null
}
```


სადაც:

id (int) - დამატებული (ან დარედაქტირებული) ოპერაციის Id,

ex (string) - ინფორმაცია შეცდომის შესახებ (ასეთის არსებობის შემთხვევაში).

saveDocCustomerReturn - საქონლის დაბრუნება მყიდველისგან

- **ფუნქცია:** saveDocCustomerReturn

  აღწერა: მყიდველისგან დაბრუნებული საქონლის შენახვა (insert, update)

  მეთოდი: POST

  გამოძახება: api/operation/saveDocCustomerReturn

Request Body:


```
{
   "id": 0,
   "date": "2019-05-01T14:25:00",
   "num_pfx": "",
   "doc_num": 0,
   "purpose": "dabruneba",
   "amount": 65.7,
   "currency": "GEL",
   "rate": 1,
   "store": 1,
   "user": 2,
   "staff": 3,
   "project": 2,
   "customer": 8,
   "is_vat": true,
   "make_entry": true,
   "pay_type": 1,
   "t_type": 1,
   "t_payer": 2,
   "w_cost": 0.5,
   "foreign": false,
   "drv_name": "მძღოლის სახელი, გვარი",
   "tr_start": "ტრანსპორტირების დაწყების ადგილი",
   "tr_end": "ტრანსპორტირების დასრულების ადგილი",
   "driver_id": "12345678910",
   "car_num": "SAK001",
   "tr_text": "მისაბმელი/გადამზიდავი",
   "products": [{
      "id": 29,
      "sub_id": 0,
      "quantity": 1,
      "price": 35.7,
      "out_id": 6547
   }, {
      "id": 3,
      "sub_id": 0,
      "quantity": 3,
      "price": 10,
      "out_id": 0
```

```
   }]
}
```

სადაც:

id (int) - ოპერაციის Id. (თუ იქმნება ახალი, გადაეცემა 0),

date (DateTime) - ოპერაციის თარიღი,

num_pfx (string[20]) - დოკუმენტის ნომრის პრეფიქსი,

num (long) - დოკუმენტის ნომერი,

purpose (string[750]) - დოკუმენტის შინაარსი (კომენტარი),

amount (decimal) - ოპერაციის ჯამური ღირებულება,

currency (string) - ვალუტის კოდი,

rate (decimal) - ვალუტის კურსი,

store (int) - საწყობის Id,

user (int) - მომხმარებლის (შემქმნელი) Id,

staff (int) - თანამშრომლის Id (0 = მომხმარებელზე მიბმული თანამშრომლის აღება),

project (int) - პროექტის Id,

customer (int) - მყიდველის Id,

is_vat (bool) - დღგ-ს შეიცავს თუ არა,

make_entry (bool) - შესრულდეს თუ არა ბუღალტრული გატარება,

pay_type (byte) - გადახდის ტიპი (0 - ნაღდი, 1 - უნაღდო, 2 - კონსიგნაცია, 3 - განვადება, 4 - ნაღდი/უნაღდო, 5 - უსასყიდლო, 6 - სხვა),

t_type (byte) - ტრანსპორტირების ტიპი (1 - საავტომობილო, 2 - სარკინიგზო, 3 - საავიაციო, 4 - სხვა, 6 - საავტომობილო უცხო ქვეყნის, 7 - გადამზიდავი საავტომობილო),

t_payer (byte) - ტრანსპორტირების ღირებულების გადამხდელი (1 - მყიდველი, 2 - გამყიდველი),

w_cost (decimal) - ტრანსპორტირების ხარჯი,

foreign (bool) - უცხო ქვეყნის მოქალაქე,

drv_name (string[50]) - მძღოლის სახელი, გვარი,

tr_start (string[200]) - ტრანსპორტირების დაწყების ადგილი,

tr_end (string[200]) - ტრანსპორტირების დასრულების ადგილი,

driver_id (string[50]) - მძღოლის პირადი ნომერი,

car_num (string[50]) - ავტომობილის ნომერი,

tr_text (string[200]) - მისაბმელი/გადამზიდავი (თუ ტრანსპორტირების ტიპი არის 7, უნდა მიეთითოს გადამზიდავის საიდენტიფიკაციო კოდი, თუ ტრანსპორტირების ტიპი არის 4 - ეთიება ტრანსპორტირების ფორმა, ხოლო საავტომობილოს შემთხვევაში - მისაბმელის ნომერი ასეთის არსებობის შემთხვევაში),

products - კოლექცია შედგება:

id (int) - საქონლის Id,

sub_id (int) - საქონლის ქვე-კოდის Id, (default=0),

quantity (decimal) - საქონლის რაოდენობა,

price (decimal) - საქონლის ერთეულის ფასი,

out_id (int) - გაყიდვის ოპერაციის Id (თუ რომელი გაყიდვის დაბრუნება ხდება კონკრეტულად, default=0).

Response:


```
{
   "id": 3,
   "ex": null
}
```


სადაც:

id (int) - დამატებული (ან დარედაქტირებული) ოპერაციის Id,

ex (string) - ინფორმაცია შეცდომის შესახებ (ასეთის არსებობის შემთხვევაში).

- **ფუნქცია:** saveDocProductIn

  აღწერა: საქონლის შესყიდვის შენახვა (insert, update)

  მეთოდი: POST

  გამოძახება: api/operation/saveDocProductIn

  Request Body:


```
{
   "id": 0,
   "date": "2019-02-18T13:00:00",
   "num_pfx": "",
   "doc_num": 0,
   "purpose": "shesyidva",
   "amount": 65.7,
   "currency": "GEL",
   "rate": 1,
   "store": 1,
   "user": 2,
   "staff": 3,
   "project": 2,
   "vendor": 2,
   "is_vat": true,
   "make_entry": true,
   "w_num": "",
   "i_num": "",
   "add_fields": [{
      "field": "usr_column_525",
      "value": "test string123"
   }],
   "products": [{
      "id": 2,
      "sub_id": 0,
      "quantity": 1,
      "price": 35.7
   }, {
      "id": 3,
      "sub_id": 0,
      "quantity": 3,
      "price": 10
   }]
}
```


  სადაც:

id (int) - ოპერაციის Id. (თუ იქმნება ახალი, გადაეცემა 0),

date (DateTime) - ოპერაციის თარიღი,

num_pfx (string[20]) - დოკუმენტის ნომრის პრეფიქსი,

num (long) - დოკუმენტის ნომერი,

purpose (string[750]) - დოკუმენტის შინაარსი (კომენტარი),

amount (decimal) - ოპერაციის ჯამური ღირებულება,

currency (string) - ვალუტის კოდი,

rate (decimal) - ვალუტის კურსი,

store (int) - საწყობის Id,

user (int) - მომხმარებლის (შემქმნელი) Id,

staff (int) - თანამშრომლის Id (0 = მომხმარებელზე მიბმული თანამშრომლის აღება),

project (int) - პროექტის Id,

vendor (int) - მომწოდებლის Id,

is_vat (bool) - დღგ-ს შეიცავს თუ არა,

make_entry (bool) - შესრულდეს თუ არა ბუღალტრული გატარება,

w_num (string[50]) - ზედნადების ნომერი,

i_num (string[50]) - ა/ფ, საბაჟო დეკლარაციის ნომერი,

add_fields - დამატებითი ველების კოლექცია სადაც:

field (string) - დამატებითი ველის დასახელება,

value (string) - დამატებითი ველის მნიშვნელობა,

products - კოლექცია შედგება:

id (int) - საქონლის Id,

sub_id (int) - საქონლის ქვე-კოდის Id, (default=0),

quantity (decimal) - საქონლის რაოდენობა,

price (decimal) - საქონლის ერთეულის ფასი.

Response:


```
{
   "id": 3,
   "ex": null
}
```


სადაც:

id (int) - დამატებული (ან დარედაქტირებული) ოპერაციის Id,

ex (string) - ინფორმაცია შეცდომის შესახებ (ასეთის არსებობის შემთხვევაში).

saveDocProductCancel - საქონლის ჩამოწერა

- **ფუნქცია:** saveDocProductCancel

  აღწერა: საქონლის ჩამოწერის შენახვა (insert, update)

  მეთოდი: POST

  გამოძახება: api/operation/saveDocProductCancel

  Request Body:

```
{
   "id": 0,
   "date": "2019-02-23T13:00:00",
   "num_pfx": "",
   "doc_num": 0,
   "purpose": "ჩამოწერა",
   "store": 1,
   "user": 2,
   "staff": 3,
   "project": 2,
   "make_entry": true,
   "add_fields": [{
      "field": "usr_column_526",
      "value": "test string123"
   }],
   "products": [{
      "id": 2,
      "sub_id": 1,
      "quantity": 1
   }, {
      "id": 3,
      "sub_id": 0,
      "quantity": 3
   }]
}
```

  სადაც:

  id (int) - ოპერაციის Id. (თუ იქმნება ახალი, გადაეცემა 0),

  date (DateTime) - ოპერაციის თარიღი,

  num_pfx (string[20]) - დოკუმენტის ნომრის პრეფიქსი,

  num (long) - დოკუმენტის ნომერი,

  purpose (string[750]) - დოკუმენტის შინაარსი (კომენტარი),

  store (int) - საწყობის Id,

  user (int) - მომხმარებლის (შემქმნელი) Id,

staff (int) - თანამშრომლის Id (0 = მომხმარებელზე მიბმული თანამშრომლის აღება),

project (int) - პროექტის Id,

make_entry (bool) - შესრულდეს თუ არა ბუღალტრული გატარება,

add_fields - დამატებითი ველების კოლექცია სადაც:

field (string) - დამატებითი ველის დასახელება,

value (string) - დამატებითი ველის მნიშვნელობა,

products - კოლექცია შედგება:

id (int) - საქონლის Id,

sub_id (int) - საქონლის ქვე-კოდის Id, (default=0),

quantity (decimal) - საქონლის რაოდენობა.

Response:


```
{
   "id": 3,
   "ex": null
}
```


სადაც:

id (int) - დამატებული (ან დარედაქტირებული) ოპერაციის Id,

ex (string) - ინფორმაცია შეცდომის შესახებ (ასეთის არსებობის შემთხვევაში).

- **ფუნქცია:** saveDocCafeOrder

  აღწერა: რესტორნის შეკვეთის შენახვა (insert, update)

  მეთოდი: POST

  გამოძახება: api/operation/saveDocCafeOrder

  Request Body:

```
{
   "id": 0,
   "date": "2019-02-28T10:00:00",
   "num_pfx": "",
   "doc_num": 0,
   "purpose": "შეკვეთა cafe",
   "amount": 9,
   "store": 2,
   "user": 2,
   "project": 2,
   "customer_name": "",
   "customer_tel": "",
   "customer_address": "",
   "products": [{
      "id": 2,
      "quantity": 1,
      "price": 3.5
   }, {
      "id": 3,
      "quantity": 3,
      "price": 1
   }],
   "services": []
}
```

  სადაც:

  id (int) - ოპერაციის Id. (თუ იქმნება ახალი, გადაეცემა 0),

  date (DateTime) - ოპერაციის თარიღი,

  num_pfx (string[20]) - დოკუმენტის ნომრის პრეფიქსი,

  num (long) - დოკუმენტის ნომერი,

  purpose (string[750]) - დოკუმენტის შინაარსი (კომენტარი),

  amount (decimal) - ოპერაციის ჯამური ღირებულება,

  store (int) - საწყობის Id,

user (int) - მომხმარებლის (შემქმნელი) Id,

project (int) - პროექტის Id,

customer_name (string[100]) - მყიდველის სახელი, გვარი,

customer_tel (string[20]) - მყიდველის ტელ.

customer_address (string[250]) - მყიდველის მისამართი.

products - კოლექცია შედგება:

id (int) - საქონლის Id,

quantity (decimal) - საქონლის რაოდენობა,

price (decimal) - საქონლის ერთეულის ფასი,

services - კოლექცია შედგება:

id (int) - გაწეული მომსახურების Id,

quantity (decimal) - გაწეული მომსახურების რაოდენობა,

price (decimal) - გაწეული მომსახურების ერთეულის ფასი.

Response:


```
{
   "id": 3,
   "ex": null
}
```


სადაც:

id (int) - დამატებული (ან დარედაქტირებული) ოპერაციის Id,

ex (string) - ინფორმაცია შეცდომის შესახებ (ასეთის არსებობის შემთხვევაში).

saveDocCustomerMoneyIn - თანხის მიღება

- **ფუნქცია:** saveDocCustomerMoneyIn

  აღწერა: მყიდველისგან მიღებული თანხის შენახვა (insert, update)

  მეთოდი: POST

  გამოძახება: api/operation/saveDocCustomerMoneyIn

  Request Body:

```
{
   "id": 0,
   "date": "2019-03-09T13:00:00",
   "num_pfx": "",
   "doc_num": 0,
   "purpose": "თანხის მიღება - მყიდველი",
   "amount": 65.7,
   "currency": "GEL",
   "rate": 1,
   "store": 1,
   "user": 1,
   "staff": 3,
   "project": 2,
   "customer": 8,
   "pay_type": 3,
   "pay_type_id": 1,
   "ref_id": 0,
   "make_entry": true,
   "add_fields": [{
      "field": "usr_column_529",
      "value": "10 ქაღალდი2"
   }]
}
```

  სადაც:

  id (int) - ოპერაციის Id. (თუ იქმნება ახალი, გადაეცემა 0),

  date (DateTime) - ოპერაციის თარიღი,

  num_pfx (string[20]) - დოკუმენტის ნომრის პრეფიქსი,

  num (long) - დოკუმენტის ნომერი,

  purpose (string[750]) - დოკუმენტის შინაარსი (კომენტარი),

  amount (decimal) - ოპერაციის ჯამური ღირებულება,

  currency (string) - ვალუტის კოდი,

  rate (decimal) - ვალუტის კურსი,

store (int) - საწყობის Id,

user (int) - მომხმარებლის (შემქმნელი) Id,

staff (int) - თანამშრომლის Id (0 = მომხმარებელზე მიბმული თანამშრომლის აღება),

project (int) - პროექტის Id,

customer (int) - მყიდველის Id,

pay_type (byte) - გადახდის სახეობა (1 - ნაღდი, 2 - ტერმინალი, 3 - საბანკო გადარიცხვა, 4 - განვადების ბანკი) ,

pay_type_id (int) - სალაროს, ტერმინალის, საბანკო ანგარიშის ან განვადების ბანკის Id არსებული FINA -ს ბაზაში (იმისდა მიხედვით რა არის არჩეული pay_type),

ref_id (int) - FINA ში არსებული ოპერაციის Id, რომლის საფუძველიცაა მოცემული თანხის მიღება.

make_entry (bool) - შესრულდეს თუ არა ბუღალტრული გატარება,

add_fields - დამატებითი ველების კოლექცია სადაც:

field (string) - დამატებითი ველის დასახელება,

value (string) - დამატებითი ველის მნიშვნელობა,

Response:


```
{
   "id": 3,
   "ex": null
}
```


სადაც:

id (int) - დამატებული (ან დარედაქტირებული) ოპერაციის Id,

ex (string) - ინფორმაცია შეცდომის შესახებ (ასეთის არსებობის შემთხვევაში).

saveDocCustomerAdvanceIn - ავანსის მიღება

- **ფუნქცია:** saveDocCustomerAdvanceIn

  აღწერა: მყიდველისგან მიღებული ავანსის შენახვა (insert, update)

  მეთოდი: POST

  გამოძახება: api/operation/saveDocCustomerAdvanceIn

  Request Body:

```
{
   "id": 0,
   "date": "2019-03-30T11:00:00",
   "num_pfx": "",
   "doc_num": 0,
   "purpose": "ავანსის მიღება - მყიდველი",
   "amount": 65.7,
   "currency": "GEL",
   "rate": 1,
   "user": 1,
   "staff": 3,
   "project": 2,
   "customer": 8,
   "pay_type": 3,
   "pay_type_id": 1,
   "overlap_type": 0,
   "overlap_amount": 0,
   "ref_id": 0,
   "make_entry": true,
   "add_fields": [{
      "field": "usr_column_531",
      "value": "10 საათი"
   }]
}
```

  სადაც:

  id (int) - ოპერაციის Id. (თუ იქმნება ახალი, გადაეცემა 0),

  date (DateTime) - ოპერაციის თარიღი,

  num_pfx (string[20]) - დოკუმენტის ნომრის პრეფიქსი,

  num (long) - დოკუმენტის ნომერი,

  purpose (string[750]) - დოკუმენტის შინაარსი (კომენტარი),

  amount (decimal) - ოპერაციის ჯამური ღირებულება,

  currency (string) - ვალუტის კოდი,

rate (decimal) - ვალუტის კურსი,

user (int) - მომხმარებლის (შემქმნელი) Id,

staff (int) - თანამშრომლის Id (0 = მომხმარებელზე მიბმული თანამშრომლის აღება),

project (int) - პროექტის Id,

customer (int) - მყიდველის Id,

pay_type (byte) - გადახდის სახეობა (1 - ნაღდი, 2 - ტერმინალი, 3 - საბანკო გადარიცხვა, 4 - განვადების ბანკი) ,

pay_type_id (int) - სალაროს, ტერმინალის, საბანკო ანგარიშის ან განვადების ბანკის Id არსებული FINA -ს ბაზაში (იმისდა მიხედვით რა არის არჩეული pay_type),

overlap_type (byte) - ავანსის დღგ-ს გამოყოფა (0 - არ გამოეყოს, 1 - ნაწილობრივ, 2 - სრულად) ,

overlap_amount (decimal) - გამოყოფოლი დღგ- ს თანხა (თუ overlap_type არის 2, მოხდება თანხის ავტომატური დაანგარიშება, თუ overlap_type არის 1 - გამოყოფოლი დღგ -ს თანხა არ უნდა აღემატებოდეს ოპერაციის ჯამურ ღირებულებაში (amount) გადმოცემული თანხიდან გამოთვლილ დღგ-ს თანხას),

ref_id (int) - FINA ში არსებული ოპერაციის Id, რომლის საფუძველიცაა მოცემული თანხის მიღება.

make_entry (bool) - შესრულდეს თუ არა ბუღალტრული გატარება,

add_fields - დამატებითი ველების კოლექცია სადაც:

field (string) - დამატებითი ველის დასახელება,

value (string) - დამატებითი ველის მნიშვნელობა,

Response:


```
{
   "id": 3,
   "ex": null
}
```


სადაც:

id (int) - დამატებული (ან დარედაქტირებული) ოპერაციის Id,

ex (string) - ინფორმაცია შეცდომის შესახებ (ასეთის არსებობის შემთხვევაში).

saveDocCustomerMoneyReturn - თანხის დაბრუნება

- **ფუნქცია:** saveDocCustomerMoneyReturn

  აღწერა: მყიდველთან დაბრუნებული თანხის შენახვა (insert, update)

  მეთოდი: POST

  გამოძახება: api/operation/saveDocCustomerMoneyReturn

  Request Body:

```
{
   "id": 0,
   "date": "2019-03-09T13:00:00",
   "num_pfx": "",
   "doc_num": 0,
   "purpose": "თანხის დაბრუნება - მყიდველი",
   "amount": 65.7,
   "currency": "GEL",
   "rate": 1,
   "store": 1,
   "user": 1,
   "staff": 3,
   "project": 2,
   "customer": 8,
   "pay_type": 1,
   "pay_type_id": 1,
   "ref_id": 0,
   "make_entry": true,
   "add_fields": [{
      "field": "usr_column_530",
      "value": "10 ქაღალდი2"
   }]
}
```

  სადაც:

  id (int) - ოპერაციის Id. (თუ იქმნება ახალი, გადაეცემა 0),

  date (DateTime) - ოპერაციის თარიღი,

  num_pfx (string[20]) - დოკუმენტის ნომრის პრეფიქსი,

  num (long) - დოკუმენტის ნომერი,

  purpose (string[750]) - დოკუმენტის შინაარსი (კომენტარი),

  amount (decimal) - ოპერაციის ჯამური ღირებულება,

  currency (string) - ვალუტის კოდი,

  rate (decimal) - ვალუტის კურსი,

store (int) - საწყობის Id,

user (int) - მომხმარებლის (შემქმნელი) Id,

staff (int) - თანამშრომლის Id (0 = მომხმარებელზე მიბმული თანამშრომლის აღება),

project (int) - პროექტის Id,

customer (int) - მყიდველის Id,

pay_type (byte) - დაბრუნების სახეობა (1 - ნაღდი, 2 - ტერმინალი, 3 - საბანკო გადარიცხვა, 4 - განვადების ბანკი) ,

pay_type_id (int) - სალაროს, ტერმინალის, საბანკო ანგარიშის ან განვადების ბანკის Id არსებული FINA -ს ბაზაში (იმისდა მიხედვით რა არის არჩეული pay_type),

ref_id (int) - FINA ში არსებული ოპერაციის Id, რომლის საფუძველიცაა მოცემული თანხის დაბრუნება.

make_entry (bool) - შესრულდეს თუ არა ბუღალტრული გატარება,

add_fields - დამატებითი ველების კოლექცია სადაც:

field (string) - დამატებითი ველის დასახელება,

value (string) - დამატებითი ველის მნიშვნელობა,

Response:


```
{
   "id": 3,
   "ex": null
}
```


სადაც:

id (int) - დამატებული (ან დარედაქტირებული) ოპერაციის Id,

ex (string) - ინფორმაცია შეცდომის შესახებ (ასეთის არსებობის შემთხვევაში).

რეპორტინგი:

getRealizesJournal - გაყიდვების ჟურნალი

- **ფუნქცია:** getRealizesJournal

აღწერა: გაყიდვების ჟურნალი (საქონლის / ძირ. საშუალების რეალიზაცია, საცალო გაყიდვა)

მეთოდი: GET

გამოძახება: api/reporting/getRealizesJournal/{date_from}/{date_to}

სადაც:

date_from (datetime) - პერიოდის საწყისი თარიღი (yyyy-MM-ddTHH:mm:ss),

date_to (datetime) - პერიოდის საბოლოო თარიღი (yyyy-MM-ddTHH:mm:ss).

Response:


```
{
   "journals": [{
      "id": 5280,
      "version": "AAAAAAAB3TM=",
      "date": "2018-12-27 18:10:48",
      "doc_num": "36",
      "waybill_num": null,
      "doc_type": 21,
      "purpose": "საქონლის რეალიზაცია ნაღდზე",
      "amount": 4.28,
      "staff_id": 3,
      "currency": "GEL",
      "customer_id": 8,
      "store_id": 1,
      "pay_type": 0
   }, {
      "id": 5282,
      "version": "AAAAAAAB3Tg=",
      "date": "2018-12-27 18:36:25",
      "doc_num": "49",
      "waybill_num": "0431440360",
      "doc_type": 23,
      "purpose": "საცალო გაყიდვა № 49",
      "amount": 4.50,
      "staff_id": 0,
      "currency": "GEL",
      "customer_id": 0,
      "store_id": 1,
      "pay_type": 0
   }],
   "ex": null
}
```

სადაც:

journal - გაყიდვების ჟურნალის კოლექციაა რომელიც შედგება:

id (int) - ოპერაციის id,

version (string) - ჩანაწერის ვერსია ბაზაში (base64 of byte[]),

date (datetime) - ოპერაციის თარიღი,

doc_num (string) - ოპერაციის ნომერი (ოპერაციის ნომრის პრეფიქსისა და ნომრის კომბინაცია),

waybill_num (string) - ზედნადების ნომერი (RS),

doc_type (int) - დოკუმენტის (ოპერაციის) ტიპი,

purpose (string) - ოპერაციის შინაარსი,

amount (decimal) - ოპერაციის თანხა,

staff_id (int) - თანამშრომლის id,

currency (string) - ვალუტა,

customer_id (int) - მყიდველის id,

store_id (int) - საწყობის id რომლიდანაც მოხდა რეალიზაცია,

pay_type (byte) - გადახდის ტიპი (0 - ნაღდი, 1 - უნაღდო, 2 - კონსიგნაცია, 3 - განვადება, 4 - ნაღდი/უნაღდო, 5 - უსასყიდლო, 6 - სხვა)

ex (string) - ინფორმაცია შეცდომის შესახებ (ასეთის არსებობის შემთხვევაში).

- **ფუნქცია:** getMovesJournal

  აღწერა: შიდა გადატანების ჟურნალი (საქონლის / ძირ. საშუალების გადატანა)

  მეთოდი: GET

  გამოძახება: api/reporting/getMovesJournal/{date_from}/{date_to}

  სადაც:

  date_from (datetime) - პერიოდის საწყისი თარიღი (yyyy-MM-ddTHH:mm:ss),

  date_to (datetime) - პერიოდის საბოლოო თარიღი (yyyy-MM-ddTHH:mm:ss).

  Response:


```
{
   "journals": [{
      "id": 14830,
      "version": "AAAAAAAG/RM=",
      "date": "2020-03-18 10:24:13",
      "doc_num": "11",
      "waybill_num": "0514054983",
      "doc_type": 20,
      "purpose": "საქონლის გადატანა № 11",
      "amount": 18.70,
      "staff_id": 2,
      "store_from_id": 1,
      "store_to_id": 2
   }, {
      "id": 14831,
      "version": "AAAAAAAG/RQ=",
      "date": "2020-03-18 10:25:25",
      "doc_num": "12",
      "waybill_num": null,
      "doc_type": 12,
      "purpose": "ძირითადი საშუალების გადაადგილება № 12",
      "amount": 5.70,
      "staff_id": 2,
      "store_from_id": 1,
      "store_to_id": 6
   }],
   "ex": null
}
```


  სადაც:

  journal - გადატანების ჟურნალის კოლექციაა რომელიც შედგება:

  id (int) - ოპერაციის id,

  version (string) - ჩანაწერის ვერსია ბაზაში (base64 of byte[]),

date (datetime) - ოპერაციის თარიღი,

doc_num (string) - ოპერაციის ნომერი (ოპერაციის ნომრის პრეფიქსისა და ნომრის კომბინაცია),

waybill_num (string) - ზედნადების ნომერი (RS),

doc_type (int) - დოკუმენტის (ოპერაციის) ტიპი,

purpose (string) - ოპერაციის შინაარსი,

amount (decimal) - ოპერაციის თანხა,

staff_id (int) - თანამშრომლის id,

store_from_id (int) - საწყობის id (გამგზავნი),

store_to_id (int) - საწყობის id (მიმღები)

ex (string) - ინფორმაცია შეცდომის შესახებ (ასეთის არსებობის შემთხვევაში).

getDocProvidedServicesJournal - გაწეული მომსახურებების ჟურნალი

- **ფუნქცია:** getDocProvidedServicesJournal

  აღწერა: გაწეული მომსახურებების ჟურნალი

  მეთოდი: GET

  გამოძახება: api/reporting/getDocProvidedServicesJournal/{date_from}/{date_to}

  სადაც:

  date_from (datetime) - პერიოდის საწყისი თარიღი (yyyy-MM-ddTHH:mm:ss),

  date_to (datetime) - პერიოდის საბოლოო თარიღი (yyyy-MM-ddTHH:mm:ss).

  Response:


```
{
   "journals": [{
      "id": 6400,
      "version": "AAAAAAACWbw=",
      "date": "2019-02-15 13:00:00",
      "doc_num": "7",
      "doc_type": 29,
      "purpose": "მომსახურების გაწევა",
      "amount": 2.00,
      "staff_id": 2,
      "currency": "GEL",
      "customer_id": 8,
      "f_status": false,
      "pay_type": 0
   }, {
      "id": 6478,
      "version": "AAAAAAAC5rk=",
      "date": "2019-03-15 12:32:18",
      "doc_num": "8",
      "doc_type": 29,
      "purpose": "მომსახურების გაწევა № 8",
      "amount": 118.00,
      "staff_id": 0,
      "currency": "GEL",
      "customer_id": 1,
      "f_status": false,
      "pay_type": 1
   }],
   "ex": null
}
```


  სადაც:

  journal - გაწეული მომსახურებების ჟურნალის კოლექციაა რომელიც შედგება:

  id (int) - ოპერაციის id,

version (string) - ჩანაწერის ვერსია ბაზაში (base64 of byte[]),

date (datetime) - ოპერაციის თარიღი,

doc_num (string) - ოპერაციის ნომერი (ოპერაციის ნომრის პრეფიქსისა და ნომრის კომბინაცია),

doc_type (int) - დოკუმენტის (ოპერაციის) ტიპი,

purpose (string) - ოპერაციის შინაარსი,

amount (decimal) - ოპერაციის თანხა,

staff_id (int) - თანამშრომლის id,

currency (string) - ვალუტა,

customer_id (int) - მყიდველის id,

f_status (bool) - გამოწერილია თუ არა ა/ფ დეკლარაცია,

pay_type (byte) - გადახდის ტიპი (0 - ნაღდი, 1 - უნაღდო, 2 - კონსიგნაცია, 3 - განვადება, 4 - ნაღდი/უნაღდო, 5 - უსასყიდლო, 6 - სხვა)

ex (string) - ინფორმაცია შეცდომის შესახებ (ასეთის არსებობის შემთხვევაში).

getDocReceivedServicesJournal - მიღებული მომსახურებების ჟურნალი

- **ფუნქცია:** getDocReceivedServicesJournal

  აღწერა: მიღებული მომსახურებების ჟურნალი

  მეთოდი: GET

  გამოძახება: api/reporting/getDocReceivedServicesJournal/{date_from}/{date_to}

  სადაც:

  date_from (datetime) - პერიოდის საწყისი თარიღი (yyyy-MM-ddTHH:mm:ss),

  date_to (datetime) - პერიოდის საბოლოო თარიღი (yyyy-MM-ddTHH:mm:ss).

  Response:


```
{
   "journals": [{
      "id": 6562,
      "version": "AAAAAAAEPh0=",
      "date": "2019-05-16 11:36:11",
      "doc_num": "4",
      "doc_type": 28,
      "purpose": "მომსახურების მიღება № 4",
      "amount": 1000.00,
      "staff_id": 0,
      "currency": "GEL",
      "vendor_id": 3,
      "pay_type": 1
   }, {
      "id": 10756,
      "version": "AAAAAAAGAy0=",
      "date": "2019-12-09 16:22:04",
      "doc_num": "5",
      "doc_type": 28,
      "purpose": "მომსახურების მიღება № 5",
      "amount": 1000.00,
      "staff_id": 0,
      "currency": "GEL",
      "vendor_id": 24,
      "pay_type": 2
   }],
   "ex": null
}
```


  სადაც:

  journal - გაწეული მომსახურებების ჟურნალის კოლექციაა რომელიც შედგება:

  id (int) - ოპერაციის id,

  version (string) - ჩანაწერის ვერსია ბაზაში (base64 of byte[]),

date (datetime) - ოპერაციის თარიღი,

doc_num (string) - ოპერაციის ნომერი (ოპერაციის ნომრის პრეფიქსისა და ნომრის კომბინაცია),

doc_type (int) - დოკუმენტის (ოპერაციის) ტიპი,

purpose (string) - ოპერაციის შინაარსი,

amount (decimal) - ოპერაციის თანხა,

staff_id (int) - თანამშრომლის id,

currency (string) - ვალუტა,

vendor_id (int) - მომწოდებლის id,

pay_type (byte) - გადახდის ტიპი (0 - ნაღდი, 1 - უნაღდო, 2 - კონსიგნაცია, 3 - განვადება, 4 - ნაღდი/უნაღდო, 5 - უსასყიდლო, 6 - სხვა)

ex (string) - ინფორმაცია შეცდომის შესახებ (ასეთის არსებობის შემთხვევაში).

getCustomersOrderJournal - მყიდველების შეკვეთების ჟურნალი

- **ფუნქცია:** getCustomersOrderJournal

  აღწერა: მყიდველებისგან მიღებული შეკვეთების ჟურნალი

  მეთოდი: GET

  გამოძახება: api/reporting/getCustomersOrderJournal/{date_from}/{date_to}

  სადაც:

  date_from (datetime) - პერიოდის საწყისი თარიღი (yyyy-MM-ddTHH:mm:ss),

  date_to (datetime) - პერიოდის საბოლოო თარიღი (yyyy-MM-ddTHH:mm:ss).

  Response:


```
{
   "journals": [{
      "id": 3121,
      "version": "AAAAAAABf5A=",
      "date": "2018-10-19 16:03:21",
      "doc_num": "5",
      "doc_type": 8,
      "purpose": "შეკვეთა მყიდველისგან № 5",
      "amount": 49.00,
      "staff_id": 4,
      "currency": "GEL",
      "customer_id": 8,
      "store_id": 1,
      "order_status": 1,
      "pay_type": 0
   }, {
      "id": 4131,
      "version": "AAAAAAACOlY=",
      "date": "2018-11-07 11:30:28",
      "doc_num": "6",
      "doc_type": 8,
      "purpose": "შეკვეთა მყიდველისგან № 6",
      "amount": 33.75,
      "staff_id": 0,
      "currency": "GEL",
      "customer_id": 8,
      "store_id": 1,
      "order_status": 3,
      "pay_type": 5
   }],
   "ex": null
}
```


  სადაც:

journal - მყიდველების შკვეთების ჟურნალის კოლექციაა რომელიც შედგება:

id (int) - ოპერაციის id,

version (string) - ჩანაწერის ვერსია ბაზაში (base64 of byte[]),

date (datetime) - ოპერაციის თარიღი,

doc_num (string) - ოპერაციის ნომერი (ოპერაციის ნომრის პრეფიქსისა და ნომრის კომბინაცია),

doc_type (int) - დოკუმენტის (ოპერაციის) ტიპი,

purpose (string) - ოპერაციის შინაარსი,

amount (decimal) - ოპერაციის თანხა,

staff_id (int) - თანამშრომლის id,

currency (string) - ვალუტა,

customer_id (int) - მყიდველის id,

store_id (int) - საწყობის id რომლშიც მოხდა საქონლის შეკვეთა,

order_status (byte) - შეკვეთის სტატუსი (1 - აქტიუდი, 2 - მიღებული, 3 - გაუქმებული),

pay_type (byte) - გადახდის ტიპი (0 - ნაღდი, 1 - უნაღდო, 2 - კონსიგნაცია, 3 - განვადება, 4 - ნაღდი/უნაღდო, 5 - უსასყიდლო, 6 - სხვა),

ex (string) - ინფორმაცია შეცდომის შესახებ (ასეთის არსებობის შემთხვევაში).

getCustomersReturnJournal - მყიდველებისგან დაბრუნებების ჟურნალი

- **ფუნქცია:** getCustomersReturnJournal

  აღწერა: მყიდველებისგან დაბრუნებული საქონლის, ძირ. საშუალებების ჟურნალი

  მეთოდი: GET

  გამოძახება: api/reporting/getCustomersReturnJournal/{date_from}/{date_to}

  სადაც:

  date_from (datetime) - პერიოდის საწყისი თარიღი (yyyy-MM-ddTHH:mm:ss),

  date_to (datetime) - პერიოდის საბოლოო თარიღი (yyyy-MM-ddTHH:mm:ss).

  Response:


```
{
   "journals": [{
      "id": 6352,
      "version": "AAAAAAACSlg=",
      "date": "2019-02-11 12:56:45",
      "doc_num": "1",
      "waybill_num": null,
      "doc_type": 9,
      "purpose": "დაბრუნება მყიდველისგან № 0",
      "amount": 7.25,
      "staff_id": 0,
      "currency": "GEL",
      "customer_id": 8,
      "store_id": 1,
      "pay_type": 1
   }, {
      "id": 6401,
      "version": "AAAAAAACaUU=",
      "date": "2019-02-15 13:00:00",
      "doc_num": "5",
      "waybill_num": null,
      "doc_type": 9,
      "purpose": "dabruneba",
      "amount": 65.7,
      "staff_id": 0,
      "currency": "GEL",
      "customer_id": 8,
      "store_id": 1,
      "pay_type": 2
   }],
   "ex": null
}
```


  სადაც:

journal - მყიდველებისგან დაბრუნებული საქონლის, ძირ. საშუალებების ჟურნალის კოლექციაა რომელიც შედგება:

id (int) - ოპერაციის id,

version (string) - ჩანაწერის ვერსია ბაზაში (base64 of byte[]),

date (datetime) - ოპერაციის თარიღი,

doc_num (string) - ოპერაციის ნომერი (ოპერაციის ნომრის პრეფიქსისა და ნომრის კომბინაცია),

waybill_num (string) - ზედნადების ნომერი (RS),

doc_type (int) - დოკუმენტის (ოპერაციის) ტიპი,

purpose (string) - ოპერაციის შინაარსი,

amount (decimal) - ოპერაციის თანხა,

staff_id (int) - თანამშრომლის id,

currency (string) - ვალუტა,

customer_id (int) - მყიდველის id,

store_id (int) - საწყობის id,

pay_type (byte) - გადახდის ტიპი (0 - ნაღდი, 1 - უნაღდო, 2 - კონსიგნაცია, 3 - განვადება, 4 - ნაღდი/უნაღდო, 5 - უსასყიდლო, 6 - სხვა),

ex (string) - ინფორმაცია შეცდომის შესახებ (ასეთის არსებობის შემთხვევაში).

getCustomersMoneyJournal - მყიდველებისგან მიღებული, დაბრუნებული თანხების და ავანსების ჟურნალი

- **ფუნქცია:** getCustomersMoneyJournal

  აღწერა: მყიდველებისგან მიღებული, დაბრუნებული თანხების და ავანსების ჟურნალი

  მეთოდი: GET

  გამოძახება: api/reporting/getCustomersMoneyJournal/{date_from}/{date_to}

  სადაც:

  date_from (datetime) - პერიოდის საწყისი თარიღი (yyyy-MM-ddTHH:mm:ss),

  date_to (datetime) - პერიოდის საბოლოო თარიღი (yyyy-MM-ddTHH:mm:ss).

  Response:


```
{
   "journals": [{
      "id": 3122,
      "version": "AAAAAAACOnw=",
      "date": "2018-10-19 16:05:04",
      "doc_num": "5",
      "doc_type": 49,
      "purpose": "ავანსის მიღება მყიდველისგან - რონალდ რეიგანი",
      "amount": 12.00,
      "staff_id": 4,
      "currency": "USD",
      "customer_id": 8,
      "pay_type": 4,
      "pay_type_id": 11
   }, {
      "id": 3123,
      "version": "AAAAAAABf5I=",
      "date": "2018-10-19 16:08:48",
      "doc_num": "5",
      "doc_type": 38,
      "purpose": "თანხის მიღება მყიდველისგან - რონალდ რეიგანი",
      "amount": 10.00,
      "staff_id": 0,
      "currency": "GEL",
      "customer_id": 8,
      "pay_type": 1,
      "pay_type_id": 1
   }],
   "ex": null
}
```


  სადაც:

journal - მყიდველებისგან მიღებული, დაბრუნებული თანხების ჟურნალის კოლექციაა რომელიც შედგება:

id (int) - ოპერაციის id,

version (string) - ჩანაწერის ვერსია ბაზაში (base64 of byte[]),

date (datetime) - ოპერაციის თარიღი,

doc_num (string) - ოპერაციის ნომერი (ოპერაციის ნომრის პრეფიქსისა და ნომრის კომბინაცია),

doc_type (int) - დოკუმენტის (ოპერაციის) ტიპი,

purpose (string) - ოპერაციის შინაარსი,

amount (decimal) - ოპერაციის თანხა,

staff_id (int) - თანამშრომლის id,

currency (string) - ვალუტა,

customer_id (int) - მყიდველის id,

pay_type (byte) - გადახდის სახეობა (1 - ნაღდი, 2 - ტერმინალი, 3 - საბანკო გადარიცხვა, 4 - განვადების ბანკი) ,

pay_type_id (int) - სალაროს, ტერმინალის, საბანკო ანგარიშის ან განვადების ბანკის Id არსებული FINA -ს ბაზაში (იმისდა მიხედვით თუ რა არის pay_type),

ex (string) - ინფორმაცია შეცდომის შესახებ (ასეთის არსებობის შემთხვევაში).

getEntriesJournal - ბუღალტრული გატარებების ჟურნალი

- **ფუნქცია:** getEntriesJournal

  აღწერა: ბუღალტრული გატარებების ჟურნალი

  მეთოდი: GET

  გამოძახება: api/reporting/getEntriesJournal/{date_from}/{date_to}

  სადაც:

  date_from (datetime) - პერიოდის საწყისი თარიღი (yyyy-MM-ddTHH:mm:ss),

  date_to (datetime) - პერიოდის საბოლოო თარიღი (yyyy-MM-ddTHH:mm:ss).

  Response:


```
{
   "journals": [{
      "id": 3121,
      "version": "AAAAAAABf5A=",
      "date": "2018-10-19 16:03:21",
      "doc_num": "5",
      "waybill_num": null,
      "doc_type": 49,
      "purpose": "ავანსის მიღება მყიდველისგან - რონალდ რეიგანი",
      "amount": 12.00,
      "staff_id": 1,
      "debit_acc": "1110.1",
      "debit_comment": "მთავარი სალარო",
      "credit_acc": "3120.8",
      "credit_comment": "რონალდ რეიგანი"
   }, {
      "id": 3121,
      "version": "AAAAAAABf5A=",
      "date": "2018-10-19 16:03:21",
      "doc_num": "5",
      "waybill_num": null,
      "doc_type": 38,
      "purpose": "ავანსის მიღება მყიდველისგან - რონალდ რეიგანი",
      "amount": 1.83,
      "staff_id": 1,
      "debit_acc": "3339",
      "debit_comment": "გადასახდელი დღგ-ს ტრანზიტული ანგარიში",
      "credit_acc": "3330",
      "credit_comment": "გადასახდელი დღგ"
   }],
   "ex": null
}
```


  სადაც:

journal - ბუღალტრული გატარებების ჟურნალის კოლექციაა რომელიც შედგება:

id (int) - ოპერაციის id,

version (string) - ჩანაწერის ვერსია ბაზაში (base64 of byte[]),

date (datetime) - ოპერაციის თარიღი,

doc_num (string) - ოპერაციის ნომერი (ოპერაციის ნომრის პრეფიქსისა და ნომრის კომბინაცია),

waybill_num (string) - ზედნადების ნომერი (RS),

doc_type (int) - დოკუმენტის (ოპერაციის) ტიპი,

purpose (string) - ოპერაციის შინაარსი,

amount (decimal) - ოპერაციის თანხა,

staff_id (int) - თანამშრომლის id,

debit_acc (string) - დებეტ ანგარიში,

debit_comment (string) - დებეტ ანგარიშის შესაბამისი აღწერა,

credit_acc (string) - კრედიტ ანგარიში,

credit_comment (string) - კრედიტ ანგარიშის შესაბამისი აღწერა,

ex (string) - ინფორმაცია შეცდომის შესახებ (ასეთის არსებობის შემთხვევაში).

getDiscountCardsJournal - ფასდაკლების ბარათების ჟურნალი

- **ფუნქცია:** getDiscountCardsJournal

  აღწერა: გაცემული ფასდაკლების ბარათების ჟურნალი

  მეთოდი: GET

  გამოძახება: api/reporting/getDiscountCardsJournal/{date_from}/{date_to}

  სადაც:

  date_from (datetime) - პერიოდის საწყისი თარიღი (yyyy-MM-ddTHH:mm:ss),

  date_to (datetime) - პერიოდის საბოლოო თარიღი (yyyy-MM-ddTHH:mm:ss).

  Response:


```
{
   "journals": [{
      "id": 8728,
      "version": "AAAAAAAFKHk=",
      "date": "2019-11-08 18:00:00",
      "doc_num": "14",
      "doc_type": 118,
      "purpose": "ფასდაკლების ბარათის გაცემა",
      "amount": 823.44,
      "staff_id": 0,
      "customer": 31,
      "store": 1,
      "card_code": "231",
      "discount_id": 1,
      "status": true
   }, {
      "id": 8733,
      "version": "AAAAAAAFKHg=",
      "date": "2019-11-08 18:00:00",
      "doc_num": "15",
      "doc_type": 118,
      "purpose": "ფასდაკლების ბარათის გაცემა",
      "amount": 823.44,
      "staff_id": 0,
      "customer": 31,
      "store": 1,
      "card_code": "2311",
      "discount_id": 2,
      "status": true
   }],
   "ex": null
}
```

სადაც:

journal - ფასდაკლების ბარათების ჟურნალის კოლექციაა რომელიც შედგება:

id (int) - ოპერაციის (ბარათის) id,

version (string) - ჩანაწერის ვერსია ბაზაში (base64 of byte[]),

date (datetime) - ოპერაციის თარიღი,

doc_num (string) - ოპერაციის ნომერი (ოპერაციის ნომრის პრეფიქსისა და ნომრის კომბინაცია),

doc_type (int) - დოკუმენტის (ოპერაციის) ტიპი,

purpose (string) - ოპერაციის შინაარსი,

amount (decimal) - მყიდველის მიერ დაგროვილი თანხა (ტრანსაზციების თანხის ჯამი, რომელშიც აღნიშნული ბარათი მონაწილეობდა),

staff_id (int) - თანამშრომლის id,

customer_id (int) - მყიდველის id,

store_id (int) - საწყობის id რომლიდანაც მოხდა ბარათის გაცემა,

card_code (string) - ბარათის კოდი,

discount_id (int) - ფასდაკლების id,

status (bool) - ბარათის სტატუსი (აქტიურია თუ არა)

ex (string) - ინფორმაცია შეცდომის შესახებ (ასეთის არსებობის შემთხვევაში).

getCustomersCycleReport - მყიდველების ბრუნვა

- **ფუნქცია:** getCustomersCycleReport

  აღწერა: მყიდელების ბრუნვის რეპორტი

  მეთოდი: GET

  გამოძახება: api/reporting/getCustomersCycleReport/{date_from}/{date_to}

  სადაც:

  date_from (datetime) - პერიოდის საწყისი თარიღი (yyyy-MM-ddTHH:mm:ss),

  date_to (datetime) - პერიოდის საბოლოო თარიღი (yyyy-MM-ddTHH:mm:ss).

  Response:

```
{
   "reports": [{
      "id": 1,
      "start_val": 2000.54,
      "in_val": 640.00,
      "out_val": 0.00,
      "end_val": 2640.54
   }, {
      "id": 6,
      "start_val": 1080.00,
      "in_val": 0.00,
      "out_val": 0.00,
      "end_val": 1080.00
   }],
   "ex": null
}
```

  სადაც:

  reports - მყიდველების ბრუნვის რეპორტის კოლექციაა რომელიც შედგება:

  id (int) - მყიდველის id,

  start_val (decimal) - ნაშთი პერიოდის დასაწყისში,

  in_val (decimal) - მიწოდებული,

  out_val (decimal) - გადახდილი,

  end_val (decimal) - ნაშთი პერიოდის ბოლოს

  ex (string) - ინფორმაცია შეცდომის შესახებ (ასეთის არსებობის შემთხვევაში).

getVendorsCycleReport - მომწოდებლების ბრუნვა

- **ფუნქცია:** getVendorsCycleReport

  აღწერა: მომწოდებლების ბრუნვის რეპორტი

  მეთოდი: GET

  გამოძახება: api/reporting/getVendorsCycleReport/{date_from}/{date_to}

  სადაც:

  date_from (datetime) - პერიოდის საწყისი თარიღი (yyyy-MM-ddTHH:mm:ss),

  date_to (datetime) - პერიოდის საბოლოო თარიღი (yyyy-MM-ddTHH:mm:ss).

  Response:

```
{
   "reports": [{
      "id": 2,
      "start_val": 2887.27,
      "in_val": 2275.70,
      "out_val": 3136.00,
      "end_val": 2026.97
   }, {
      "id": 3,
      "start_val": 4584.28,
      "in_val": 620.00,
      "out_val": 323.27,
      "end_val": 4881.01
   }],
   "ex": null
}
```

  სადაც:

  reports - მომწოდებლების ბრუნვის რეპორტის კოლექციაა რომელიც შედგება:

  id (int) - მომწოდებლის id,

  start_val (decimal) - ნაშთი პერიოდის დასაწყისში,

  in_val (decimal) - მიღებული,

  out_val (decimal) - გადახდილი,

  end_val (decimal) - ნაშთი პერიოდის ბოლოს

  ex (string) - ინფორმაცია შეცდომის შესახებ (ასეთის არსებობის შემთხვევაში).

getProductsLastInReport - საქონლის ბოლო მიღების ინფორმაცია

- **ფუნქცია:** getProductsLastInReport

  აღწერა: საქონლის ბოლო მიღების ინფორმაცია

  მეთოდი: GET

  გამოძახება: api/reporting/getProductsLastInReport/{date_from}/{date_to}

  სადაც:

  date_from (datetime) - პერიოდის საწყისი თარიღი (yyyy-MM-ddTHH:mm:ss),

  date_to (datetime) - პერიოდის საბოლოო თარიღი (yyyy-MM-ddTHH:mm:ss).

  Response:

```
{
   "reports": [{
      "id": 1,
      "quantity": 2.00,
      "price": 118.00,
      "vendor_id": 25,
      "currency": "GEL",
      "rate": 1.00,
      "waybill_date": "2019-04-01T16:12:00",
      "waybill_num": "",
      "purpose": "საქონლის მიღება № 10"
   }, {
      "id": 2,
      "quantity": 1.00,
      "price": 35.70,
      "vendor_id": 2,
      "currency": "GEL",
      "rate": 1.00,
      "waybill_date": "2019-02-18T13:00:00",
      "waybill_num": "",
      "purpose": "shesyidva"
   }],
   "ex": null
}
```

  სადაც:

  reports - საქონლის ბოლო მიღების ინფორმაციის რეპორტის კოლექციაა რომელიც შედგება:

  id (int) - საქონლის id,

  quantity (decimal) - რაოდენობა

  price (decimal) - ფასი,

  vendor_id (int) - მომწოდებლის id,

  currency (string) - ვალუტა,

rate (decimal) - ვალუტის კურსი,

waybill_date (string) - ზედნადების თარიღი,

waybill_num (string) - ზედნადების RS ნომერი,

purpose (string) - ოპერაციის შინაარსი

ex (string) - ინფორმაცია შეცდომის შესახებ (ასეთის არსებობის შემთხვევაში).