Errors

Errors are returned as follows:

Common:

{

    “status”: “error”,

    “errors”: [

      “Some error”,

    ],

}

Forms (errors in fields):

{

    “status”: “error”,

    “errors”: [...],

    “fields”: {

      “field1”: [...],

      “field2”: [...],

      ...

    }

}

Authentication

We have 2 different roles of users: **admin** and **customer**. Both can sign in using common API. But only customers are registered using API. The first admin account should be created manually at PostgreSQL or a script implemented at the server side. Other admin accounts can be created by the first one using admin panel.

Note: **Everyone** means authorization is not required for this method.

POST /api/users/signup/

Customer registration

Access level: everyone

Params (application/x-www-form-urlencoded):

●     login (required)

●     password (required)

●     first\_name (required)

●     last\_name (required)

●     email (required)

Returns JSON:

{

    “status”: “ok”

}

POST /api/users/signin/

User authentication

Access level: everyone

Params (application/x-www-form-urlencoded):

●     login (required)

●     password (required)

Returns Header:

●     “Authorization” -> “Bearer <token>”

Returns JSON:

{

"id": <id>,

"login": <username>,

"email": <email>,

"first\_name": <first>,

"last\_name": <last>,

"role": [

<role1>, <role2>

],

"token": <token>

}

Token is available for 1 hour. [debug](https://jwt.io/#debugger-io) Example of token:

eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJhZG1pbiIsImlzcyI6InRpZXRva29uZWthdXBwYSIsImV4cCI6MTUwNjg4NTI3OSwiaWF0IjoxNTA2ODgxNjc5fQ.P1rz4smbDklgZjBPByRXh5mUcsZf1YO2bSN0MnlGo6g

POST /api/users/logout/

Exit profile

Access level: customer/admin

Возвращает JSON:

{

    “status”: “ok”

}

Component resources

Methods listed here are the same for /**disks**, **/cases**, /**gpus**, /**motherboards**, /**processors**, /**psus**, /**rams**.

GET /api/disks/

Gets all disks available in the database.

Access level: everyone

Params:

●     min\_price (int, default value = 0)

●     max\_price (int, default value = 1000000)

●     price\_units (String {EUR/USD})

Returns the list of Disk objects in JSON.

GET /api/disks/<disk\_id>/

Retrieves the disk with id=<disk\_id>

Access level: everyone

Returns the Disk object in JSON

GET /api/disks/<disk\_id>/comments/

Retrieves all comments for the disk with id=<disk\_id>

Access level: everyone

Returns the list of Comment objects in JSON

POST /api/disks/<disk\_id>/comments/

Posts a new comment for the disk with id=<disk\_id>

Access level: customer

Params (path parameter):

●     contents

●     ~~username~~ (will not be required soon because we can retrieve user from SecureContext)

Returns the newly created Comment object in JSON.

GET /api/disks/<disk\_id>/comments/<comment\_id>/

Retrieves the comment with id=<comment\_id> for the disk with id=<disk\_id>

Access level: everyone

Returns the Comment object in JSON

DELETE /api/disks/<disk\_id>/comments/<comment\_id>/

Deleted the comment with id=<comment\_id>

Access level: admin / customer (which is the owner of the comment with id=<comment\_id>)

Returns JSON:

{

    “status”: “ok”

}

Order resources

GET /api/orders/

Retrieves all orders of the customer

Access level: customer

Returns the list of Order objects in JSON

POST /api/orders/

Creates a new order for the customer

Access level: customer

Params (path parameter):

●     notes (String describing some details, not required)

Data:

●     List of Component objects with explicit “@type” and “id” fields

Example of json data body:

● [ {"id": 12, "@type": "Disk"}, {"id": 15, "@type": Disk}, {"id": 50, "Case"} ]

Returns the newly created Order object in JSON.

Admin panel

Access level: admin

For each model of our application we have the following CRUD methods. Access to these methods is provided only for admins.

GET /api/admin/entities/

POST /api/admin/entities/

GET /api/admin/entities/<entity\_id>/

PUT /api/admin/entities/<entity\_id>/

DELETE /api/admin/entities/<entity\_id>/