

---

# **StoreKeeper Documentation**

***Release 0.6.0-beta***

**Andras Tim**

March 18, 2016



<b>1</b>	<b>Features</b>	<b>3</b>
<b>2</b>	<b>Demo site</b>	<b>5</b>
<b>3</b>	<b>Languages</b>	<b>7</b>
<b>4</b>	<b>The Guide</b>	<b>9</b>
4.1	Setup . . . . .	9
4.2	Upgrade . . . . .	9
4.3	Client . . . . .	10
4.4	Server . . . . .	10
<b>5</b>	<b>Indices and tables</b>	<b>65</b>
	<b>HTTP Routing Table</b>	<b>67</b>



StoreKeeper is an open source, multilingual warehouse/store management software

**Warning:** *Project is under development!*



---

# Features

---

The following features are available:

- User login, logout
- Manage items and its barcodes, units, vendors
- Add/remove items in store
- Able to use barcode reader for collecting items
- Can use barcode printer to create labels

The following features are only available via API now:

- Manage users, handle permissions
- Can make acquisition and stocktaking
- Can crate works and its customers (handle outbound, returned items)





---

### Demo site

---

**You can test the latest development version on our demo server:** <http://storekeeper-demo.dras.hu/storekeeper>  
Default username and password: **admin / admin**



---

# Languages

---

StoreKeeper is building in multilingual approach, therefore it can easily translate to a new language.

- You should create a new `.po` file in your language, then put to `client/po/` directory and create a pull request.
- Or use *Transifex* for online translation: <https://www.transifex.com/projects/p/storekeeper/>



## 4.1 Setup

### 4.1.1 Prepare & Run

All commands can run with `package.sh` scripts what you want. All of these scripts has `--help` argument for available getting some info about the current module.

1. Clone repo, or download & extract a release `.tar.gz` file
2. Install all dependencies: `./package.sh -p install`
3. Check the `config/config.yml` for database and other settings
4. Prepare database: `./package.sh upgrade_database`
5. Start server: `./package.sh start`

Now, you can open the WebUI: <http://localhost:8000/storekeeper>

### 4.1.2 Details

Basically, the `install` command consists of `preinstall` and `postinstall` parts.

- `preinstall` checks/prepares system components for `postinstall` and `start`. This command has only one dependency, an **DEB** based system (for *apt-get install*).
- `postinstall` checks/prepares external dependencies (e.g. Python, Bower modules).

You can modify installing method with this arguments:

- `--global` makes changes on system instead of virtual environments.
- `--production` installs dependencies for production running only (e.g. did not install unit test framework)

## 4.2 Upgrade

The upgrade process is under construction, but there are the main steps:

1. Update source code with **git** / extract release `.tar.gz` from GitHub.
2. Update libs and others with `./package.sh -p install`.
3. Follow up config changes based on `config/config.default.yml`.
4. Upgrade database scheme with `./package.sh upgrade_database`.

---

**Note:** Proper, seamless upgrade process will be supported between the stable versions!

---

## 4.2.1 Upgrade from v0.2.1 to v0.3.0

Changed the database migration framework in **v0.3.0**, therefore have to make some changes by hand:

1. Do common upgrade scenario step-by-step **without** upgrading database (*step 4*)
2. Remove `server/db_repository` directory
3. Run the following SQL commands:

```
ALTER TABLE item ALTER COLUMN purchase_price SET DEFAULT 0;

DROP TABLE migrate_version;
CREATE TABLE alembic_version (version_num character varying(32) NOT NULL);
-- ALTER TABLE public.alembic_version OWNER TO sql_user_of_storekeeper;
INSERT INTO alembic_version VALUES ('305c2b0084f');
```

4. Now, upgrade database scheme with `./package.sh upgrade_database`.

---

**Note:** Run this commands as StoreKeeper's SQL user or use `ALTER TABLE` to set owner of the new table.

---

## 4.3 Client

The UI of **StoreKeeper** will be implemented in [AngularStrap](#).

## 4.4 Server

The backend of **StoreKeeper** is based on [Flask](#) and used several plugins of it.

### 4.4.1 Interfaces

#### Admin page

**StoreKeeper** has admin interface (with [Flask-Admin](#)), where you can manage database tables and static files directly.















You can reach this on: <http://localhost:8000/<name>/admin>

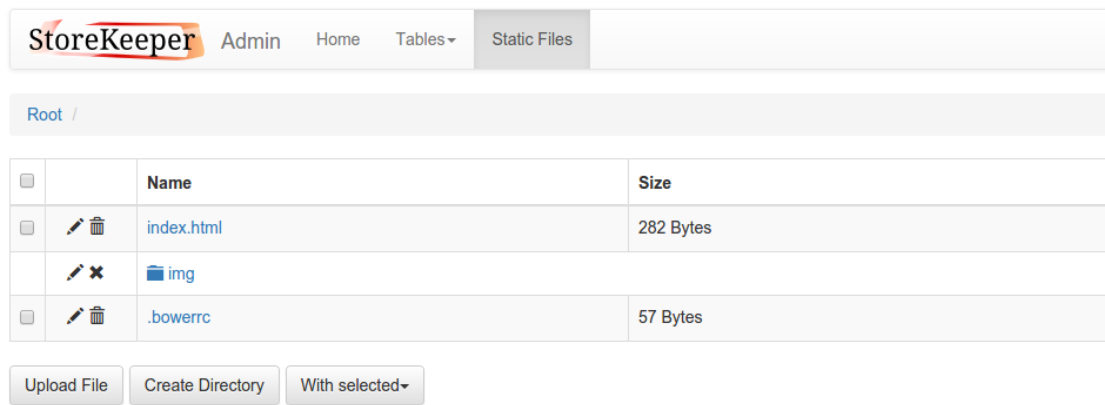
---

**Note:** Admin page is not available in in production mode.

---

## Interface

StoreKeeper Admin Home Tables Static Files						
List (7)	Create	With selected▼				
<input type="checkbox"/>		Vendor	Unit		Article Number	Quantity
<input type="checkbox"/>	 	Foo	pcs		67845234	12
<input type="checkbox"/>	 	Pipe Maker	m		123134	12
<input type="checkbox"/>	 	Pipe Maker	m		6666	12
<input type="checkbox"/>	 	Pipe Maker	m		6666	12
<input type="checkbox"/>	 	Pipe Maker	m		6666	12
<input type="checkbox"/>	 	Pipe Maker	m		6666	12
<input type="checkbox"/>	 	Pipe Maker	m	Water pipe7	6666	12



## RPC API

**StoreKeeper** uses [HATEOAS](#) RCP API for communicate server and client side each other. The URLs are prefixed with name of application what you can customize in `server/config.yml`.

*Example URL:*

```
http://localhost:8000/<name>/api/<command>
```

## Endpoints

**Acquisitions** API endpoint for manage acquisitions.

### Data management

#### `/api/acquisitions`

**GET** `/storekeeper/api/acquisitions`

List acquisitions

#### Status Codes

- 200 OK – no error
- 401 Unauthorized – user was not logged in

**Example request:**

```
GET /storekeeper/api/acquisitions HTTP/1.1
Host: localhost:8000
Content-Type: application/json
```



**Example response:**

```

HTTP/1.0 200 OK
Content-Type: application/json

[
  {
    "comment": "Maybe missing some items",
    "id": 1,
    "timestamp": "2016-03-18T07:58:11.557018+00:00"
  },
  {
    "comment": null,
    "id": 2,
    "timestamp": "2016-03-18T07:58:11.557121+00:00"
  }
]

```

**POST /storekeeper/api/acquisitions**

Create acquisition

**Status Codes**

- 201 Created – no error
- 401 Unauthorized – user was not logged in
- 422 Unprocessable Entity – there is wrong type / missing field

**Example request:**

```

POST /storekeeper/api/acquisitions HTTP/1.1
Host: localhost:8000
Content-Type: application/json

{
  "comment": "Maybe missing some items"
}

```

**Example response:**

```

HTTP/1.0 201 CREATED
Content-Type: application/json

{
  "comment": "Maybe missing some items",
  "id": 1,
  "timestamp": "2016-03-18T07:58:11.557018+00:00"
}

```

**/api/acquisitions/<id>****GET /storekeeper/api/acquisitions/ (int: id)**

Get acquisition

**Parameters**

- **id** – ID of selected acquisition for get

**Status Codes**

- 200 OK – no error
- 401 Unauthorized – user was not logged in
- 404 Not Found – there is no acquisition

**Example request:**

```
GET /storekeeper/api/acquisitions/1 HTTP/1.1
Host: localhost:8000
Content-Type: application/json
```

### Example response:

```
HTTP/1.0 200 OK
Content-Type: application/json

{
  "comment": "Maybe missing some items",
  "id": 1,
  "timestamp": "2016-03-18T07:58:11.557018+00:00"
}
```

**PUT** /storekeeper/api/acquisitions/ (int: *id*)  
Update acquisition

### Parameters

- **id** – ID of selected acquisition for put

### Status Codes

- 200 OK – no error
- 401 Unauthorized – user was not logged in
- 404 Not Found – there is no acquisition
- 422 Unprocessable Entity – there is wrong type / missing field

### Example request:

```
PUT /storekeeper/api/acquisitions/1 HTTP/1.1
Host: localhost:8000
Content-Type: application/json

{
  "comment": "A box has been damaged"
}
```

### Example response:

```
HTTP/1.0 200 OK
Content-Type: application/json

{
  "comment": "A box has been damaged",
  "id": 1,
  "timestamp": "2016-03-18T07:58:11.557018+00:00"
}
```

**DELETE** /storekeeper/api/acquisitions/ (int: *id*)  
Delete acquisition

### Parameters

- **id** – ID of selected acquisition for delete

### Status Codes

- 200 OK – no error
- 401 Unauthorized – user was not logged in
- 404 Not Found – there is no acquisition

### Example request:

```
DELETE /storekeeper/api/acquisitions/1 HTTP/1.1
Host: localhost:8000
Content-Type: application/json
```

**Example response:**

```
HTTP/1.0 200 OK
Content-Type: application/json

null
```

**/api/acquisitions/<id>/items****GET /storekeeper/api/acquisitions/ (int: id) /items**

List acquisition items

**Parameters**

- **id** – ID of acquisition

**Status Codes**

- **200 OK** – no error
- **401 Unauthorized** – user was not logged in
- **404 Not Found** – there is no item

**Example request:**

```
GET /storekeeper/api/acquisitions/1/items HTTP/1.1
Host: localhost:8000
Content-Type: application/json
```

**Example response:**

```
HTTP/1.0 200 OK
Content-Type: application/json

[
  {
    "id": 1,
    "item": {
      "article_number": "FO213546",
      "id": 2,
      "location": "B02/01",
      "name": "Pipe",
      "purchase_price": 0.0,
      "quantity": 0.0,
      "unit": {
        "id": 1,
        "unit": "m"
      },
      "vendor": {
        "id": 2,
        "name": "Star Shop Ltd."
      },
      "warning_quantity": 0.0
    },
    "quantity": 132.4
  },
  {
    "id": 2,
    "item": {
      "article_number": "SK132465",
      "id": 1,
```

```
    "location": "A12/12",
    "name": "Spray",
    "purchase_price": 60.4,
    "quantity": 0.0,
    "unit": {
      "id": 2,
      "unit": "pcs"
    },
    "vendor": {
      "id": 1,
      "name": "Heavy Duty Ltd."
    },
    "warning_quantity": 4.0
  },
  "quantity": 32.1
}
```

**POST** /storekeeper/api/acquisitions/ (int: id) /items  
Create acquisition item

#### Parameters

- **id** – ID of acquisition

#### Status Codes

- 201 Created – no error
- 401 Unauthorized – user was not logged in
- 404 Not Found – there is no item
- 422 Unprocessable Entity – there is wrong type / missing field / can not add one item twice

#### Example request:

```
POST /storekeeper/api/acquisitions/1/items HTTP/1.1
Host: localhost:8000
Content-Type: application/json

{
  "item": {
    "article_number": "FO213546",
    "id": 2,
    "location": "B02/01",
    "name": "Pipe",
    "purchase_price": 0.0,
    "quantity": 0.0,
    "unit": {
      "id": 1,
      "unit": "m"
    },
    "vendor": {
      "id": 2,
      "name": "Star Shop Ltd."
    },
    "warning_quantity": 0.0
  },
  "quantity": 132.4
}
```

#### Example response:

```

HTTP/1.0 201 CREATED
Content-Type: application/json

{
  "id": 1,
  "item": {
    "article_number": "FO213546",
    "id": 2,
    "location": "B02/01",
    "name": "Pipe",
    "purchase_price": 0.0,
    "quantity": 0.0,
    "unit": {
      "id": 1,
      "unit": "m"
    },
    "vendor": {
      "id": 2,
      "name": "Star Shop Ltd."
    },
    "warning_quantity": 0.0
  },
  "quantity": 132.4
}

```

/api/acquisitions/<id>/items/<item\_id>

GET /storekeeper/api/acquisitions/ (int: id) /items/  
int: item\_id Get acquisition item

#### Parameters

- **id** – ID of acquisition
- **item\_id** – ID of selected acquisition item for get

#### Status Codes

- 200 OK – no error
- 401 Unauthorized – user was not logged in
- 404 Not Found – there is no acquisition item

#### Example request:

```

GET /storekeeper/api/acquisitions/1/items/1 HTTP/1.1
Host: localhost:8000
Content-Type: application/json

```

#### Example response:

```

HTTP/1.0 200 OK
Content-Type: application/json

{
  "id": 1,
  "item": {
    "article_number": "FO213546",
    "id": 2,
    "location": "B02/01",
    "name": "Pipe",
    "purchase_price": 0.0,
    "quantity": 0.0,
    "unit": {
      "id": 1,

```

```
    "unit": "m"
  },
  "vendor": {
    "id": 2,
    "name": "Star Shop Ltd."
  },
  "warning_quantity": 0.0
},
"quantity": 132.4
}
```

**PUT** `/storekeeper/api/acquisitions/ (int: id) /items/`  
**int: item\_id** Update acquisition item

#### Parameters

- **id** – ID of acquisition
- **item\_id** – ID of selected acquisition item for get

#### Status Codes

- 200 OK – no error
- 401 Unauthorized – user was not logged in
- 404 Not Found – there is no acquisition item
- 422 Unprocessable Entity – there is wrong type / missing field / can not add one item twice

#### Example request:

```
PUT /storekeeper/api/acquisitions/1/items/1 HTTP/1.1
Host: localhost:8000
Content-Type: application/json

{
  "item": {
    "article_number": "FO213546",
    "id": 2,
    "location": "B02/01",
    "name": "Pipe",
    "purchase_price": 0.0,
    "quantity": 0.0,
    "unit": {
      "id": 1,
      "unit": "m"
    },
    "vendor": {
      "id": 2,
      "name": "Star Shop Ltd."
    },
    "warning_quantity": 0.0
  },
  "quantity": 132.4
}
```

#### Example response:

```
HTTP/1.0 200 OK
Content-Type: application/json

{
  "id": 1,
  "item": {
    "article_number": "FO213546",
```

```

    "id": 2,
    "location": "B02/01",
    "name": "Pipe",
    "purchase_price": 0.0,
    "quantity": 0.0,
    "unit": {
      "id": 1,
      "unit": "m"
    },
    "vendor": {
      "id": 2,
      "name": "Star Shop Ltd."
    },
    "warning_quantity": 0.0
  },
  "quantity": 132.4
}

```

**DELETE** /storekeeper/api/acquisitions/ (int: *id*) /items/  
 int: *item\_id* Delete acquisition item

#### Parameters

- **id** – ID of acquisition
- **item\_id** – ID of selected acquisition item for get

#### Status Codes

- 200 OK – no error
- 401 Unauthorized – user was not logged in
- 404 Not Found – there is no acquisition item

#### Example request:

```

DELETE /storekeeper/api/acquisitions/1/items/1 HTTP/1.1
Host: localhost:8000
Content-Type: application/json

```

#### Example response:

```

HTTP/1.0 200 OK
Content-Type: application/json

null

```

**Barcodes** API endpoint for fetching barcodes.

#### Data management

##### /api/barcodes

**GET** /storekeeper/api/barcodes

List barcodes items

#### Status Codes

- 200 OK – no error
- 401 Unauthorized – user was not logged in

#### Example request:

```
GET /storekeeper/api/barcodes HTTP/1.1
Host: localhost:8000
Content-Type: application/json
```

### Example response:

```
HTTP/1.0 200 OK
Content-Type: application/json

[
  {
    "barcode": "SKT98601",
    "id": 1,
    "item_id": 1,
    "main": true,
    "master": true,
    "quantity": 32.7
  },
  {
    "barcode": "9843-184125",
    "id": 2,
    "item_id": 1,
    "main": false,
    "master": false,
    "quantity": 1.5
  }
]
```

**Config** API endpoint for getting client related settings from server config.

### Getting config values

#### /api/config

**GET** /storekeeper/api/config

Get server settings

#### Status Codes

- 200 OK – no error

### Example request:

```
GET /storekeeper/api/config HTTP/1.1
Host: localhost:8000
Content-Type: application/json
```

### Example response:

```
HTTP/1.0 200 OK
Content-Type: application/json

{
  "app_name": "storekeeper",
  "app_title": "StoreKeeper",
  "currency": "$",
  "debug": false,
  "forced_language": null
}
```

**Customers** API endpoint for manage customers.



## Data management

### /api/customers

#### GET /storekeeper/api/customers

List customers

##### Status Codes

- 200 OK – no error
- 401 Unauthorized – user was not logged in

##### Example request:

```
GET /storekeeper/api/customers HTTP/1.1
Host: localhost:8000
Content-Type: application/json
```

##### Example response:

```
HTTP/1.0 200 OK
Content-Type: application/json

[
  {
    "id": 1,
    "name": "Big Customer Ltd."
  },
  {
    "id": 2,
    "name": "Buy Everything Co."
  }
]
```

#### POST /storekeeper/api/customers

Create customer

##### Status Codes

- 201 Created – no error
- 401 Unauthorized – user was not logged in
- 422 Unprocessable Entity – there is wrong type / missing field / customer is already exist

##### Example request:

```
POST /storekeeper/api/customers HTTP/1.1
Host: localhost:8000
Content-Type: application/json

{
  "name": "Big Customer Ltd."
}
```

##### Example response:

```
HTTP/1.0 201 CREATED
Content-Type: application/json

{
  "id": 1,
  "name": "Big Customer Ltd."
}
```

**/api/customers/<id>**

**GET** /storekeeper/api/customers/ (int: id)

Get customer

#### Parameters

- **id** – ID of selected customer for get

#### Status Codes

- **200 OK** – no error
- **401 Unauthorized** – user was not logged in
- **404 Not Found** – there is no customer

#### Example request:

```
GET /storekeeper/api/customers/1 HTTP/1.1
Host: localhost:8000
Content-Type: application/json
```

#### Example response:

```
HTTP/1.0 200 OK
Content-Type: application/json

{
  "id": 1,
  "name": "Big Customer Ltd."
}
```

**PUT** /storekeeper/api/customers/ (int: id)

Update customer

#### Parameters

- **id** – ID of selected customer for put

#### Status Codes

- **200 OK** – no error
- **401 Unauthorized** – user was not logged in
- **404 Not Found** – there is no customer
- **422 Unprocessable Entity** – there is wrong type / missing field / customer is already exist

#### Example request:

```
PUT /storekeeper/api/customers/1 HTTP/1.1
Host: localhost:8000
Content-Type: application/json

{
  "name": "new_foo"
}
```

#### Example response:

```
HTTP/1.0 200 OK
Content-Type: application/json

{
  "id": 1,
  "name": "new_foo"
}
```

**DELETE /storekeeper/api/customers/** (int: *id*)

Delete customer

**Parameters**

- **id** – ID of selected customer for delete

**Status Codes**

- 200 OK – no error
- 401 Unauthorized – user was not logged in
- 404 Not Found – there is no customer

**Example request:**

```
DELETE /storekeeper/api/customers/1 HTTP/1.1
Host: localhost:8000
Content-Type: application/json
```

**Example response:**

```
HTTP/1.0 200 OK
Content-Type: application/json

null
```

**Error** API endpoint for getting client side errors to alerting administrator.**Push error messages****/api/error****POST /storekeeper/api/error**

Receive client side errors and forward to logfile / email / syslog (what specified in config)

**Status Codes**

- 201 Created – no error
- 401 Unauthorized – user was not logged in
- 422 Unprocessable Entity – there is wrong type / missing field

**Example request:**

```
POST /storekeeper/api/error HTTP/1.1
Host: localhost:8000
Content-Type: application/json

{
  "message": "foo is not defined",
  "name": "ReferenceError",
  "stack": "ReferenceError: foo is not defined\n    at Scope.printLabel (http://.../storekeeper/...)"
}
```

**Example response:**

```
HTTP/1.0 201 CREATED
Content-Type: application/json

null
```

**Items** API endpoint for manage items.

### Data management

#### /api/items

##### GET /storekeeper/api/items

List items

##### Status Codes

- 200 OK – no error
- 401 Unauthorized – user was not logged in

##### Example request:

```
GET /storekeeper/api/items HTTP/1.1
Host: localhost:8000
Content-Type: application/json
```

##### Example response:

```
HTTP/1.0 200 OK
Content-Type: application/json

[
  {
    "article_number": "SK132465",
    "id": 1,
    "location": "A12/12",
    "name": "Spray",
    "purchase_price": 60.4,
    "quantity": 0.0,
    "unit": {
      "id": 2,
      "unit": "pcs"
    },
    "vendor": {
      "id": 1,
      "name": "Heavy Duty Ltd."
    },
    "warning_quantity": 4.0
  },
  {
    "article_number": "FO213546",
    "id": 2,
    "location": "B02/01",
    "name": "Pipe",
    "purchase_price": 0.0,
    "quantity": 0.0,
    "unit": {
      "id": 1,
      "unit": "m"
    },
    "vendor": {
      "id": 2,
      "name": "Star Shop Ltd."
    },
    "warning_quantity": 0.0
  }
]
```

##### POST /storekeeper/api/items

Create item

##### Status Codes

- 201 Created – no error

- 401 Unauthorized – user was not logged in
- 422 Unprocessable Entity – there is wrong type / missing field

**Example request:**

```
POST /storekeeper/api/items HTTP/1.1
Host: localhost:8000
Content-Type: application/json

{
  "article_number": "sk132465",
  "location": "A12/12",
  "name": "Spray",
  "purchase_price": 60.4,
  "unit": {
    "id": 2,
    "unit": "pcs"
  },
  "vendor": {
    "id": 1,
    "name": "Heavy Duty Ltd."
  },
  "warning_quantity": 4.0
}
```

**Example response:**

```
HTTP/1.0 201 CREATED
Content-Type: application/json

{
  "article_number": "SK132465",
  "id": 1,
  "location": "A12/12",
  "name": "Spray",
  "purchase_price": 60.4,
  "quantity": 0.0,
  "unit": {
    "id": 2,
    "unit": "pcs"
  },
  "vendor": {
    "id": 1,
    "name": "Heavy Duty Ltd."
  },
  "warning_quantity": 4.0
}
```

**/api/items/search****GET /storekeeper/api/items/search**

Search in items and barcodes

**Parameters**

- **barcodes** – Filter for master barcodes [0=False (default), 1=True]
- **expression** – Query string (search in barcode, item name, article number)
- **items** – Filter for items [0=False (default), 1=True]
- **limit** – Limit of result set [default: 6]

**Status Codes**

- 200 OK – no error

- 401 Unauthorized – user was not logged in

**Example request:**

```
GET /storekeeper/api/items/search?expression=sk&limit=6&barcodes=1&items=1 HTTP/1.1
Host: localhost:8000
Content-Type: application/json
```

**Example response:**

```
HTTP/1.0 200 OK
Content-Type: application/json

[
  {
    "barcode": "SK586936",
    "item_id": 326,
    "quantity": 1,
    "type": "barcode"
  },
  {
    "article_number": "SE180826",
    "item_id": 309,
    "name": "Skate",
    "type": "item",
    "vendor": "Import"
  }
]
```

**/api/items/<id>**

**GET /storekeeper/api/items/ (int: id)**

Get item

**Parameters**

- **id** – ID of selected item for get

**Status Codes**

- 200 OK – no error
- 401 Unauthorized – user was not logged in
- 404 Not Found – there is no item

**Example request:**

```
GET /storekeeper/api/items/1 HTTP/1.1
Host: localhost:8000
Content-Type: application/json
```

**Example response:**

```
HTTP/1.0 200 OK
Content-Type: application/json

{
  "article_number": "SK132465",
  "id": 1,
  "location": "A12/12",
  "name": "Spray",
  "purchase_price": 60.4,
  "quantity": 0.0,
  "unit": {
    "id": 2,
    "unit": "pcs"
  }
}
```

```

    },
    "vendor": {
      "id": 1,
      "name": "Heavy Duty Ltd."
    },
    "warning_quantity": 4.0
  }
}

```

**PUT** /storekeeper/api/items/ (int: id)

Update item

#### Parameters

- **id** – ID of selected item for put

#### Status Codes

- 200 OK – no error
- 401 Unauthorized – user was not logged in
- 404 Not Found – there is no item
- 422 Unprocessable Entity – there is wrong type / missing field

#### Example request:

```

PUT /storekeeper/api/items/1 HTTP/1.1
Host: localhost:8000
Content-Type: application/json

{
  "article_number": "sk132465",
  "location": "A12/12",
  "name": "Spray",
  "purchase_price": 60.4,
  "unit": {
    "id": 2,
    "unit": "pcs"
  },
  "vendor": {
    "id": 1,
    "name": "Heavy Duty Ltd."
  },
  "warning_quantity": 4.0
}

```

#### Example response:

```

HTTP/1.0 200 OK
Content-Type: application/json

{
  "article_number": "SK132465",
  "id": 1,
  "location": "A12/12",
  "name": "Spray",
  "purchase_price": 60.4,
  "quantity": 0.0,
  "unit": {
    "id": 2,
    "unit": "pcs"
  },
  "vendor": {
    "id": 1,
    "name": "Heavy Duty Ltd."
  },
}

```

```
"warning_quantity": 4.0
}
```

**DELETE** /storekeeper/api/items/ (int: id)

Delete item

### Parameters

- **id** – ID of selected item for delete

### Status Codes

- 200 OK – no error
- 401 Unauthorized – user was not logged in
- 404 Not Found – there is no item

### Example request:

```
DELETE /storekeeper/api/items/1 HTTP/1.1
Host: localhost:8000
Content-Type: application/json
```

### Example response:

```
HTTP/1.0 200 OK
Content-Type: application/json

null
```

/api/items/<id>/barcodes

**GET** /storekeeper/api/items/ (int: id) /barcodes

List barcodes.

### Parameters

- **id** – ID of item

### Status Codes

- 200 OK – no error
- 401 Unauthorized – user was not logged in
- 404 Not Found – there is no item

### Example request:

```
GET /storekeeper/api/items/1/barcodes HTTP/1.1
Host: localhost:8000
Content-Type: application/json
```

### Example response:

```
HTTP/1.0 200 OK
Content-Type: application/json

[
  {
    "barcode": "SKT52680",
    "id": 1,
    "main": true,
    "master": true,
    "quantity": 32.7
  },
  {
    "barcode": "9843-184125",
```



```

    "id": 2,
    "main": false,
    "master": false,
    "quantity": 1.5
  }
]

```

**POST /storekeeper/api/items/ (int: id) /barcodes**

Create barcode (if missing barcode then server will generate one)

#### Parameters

- **id** – ID of item

#### Status Codes

- 201 Created – no error
- 401 Unauthorized – user was not logged in
- 404 Not Found – there is no item
- 422 Unprocessable Entity – there is wrong type / missing field / can not add one barcode twice / can not generate unique new barcode / can not set non-main barcode as master barcode / can not set more than one master barcode to an item

#### Example request:

```

POST /storekeeper/api/items/1/barcodes HTTP/1.1
Host: localhost:8000
Content-Type: application/json

{
  "barcode": "SKT52680",
  "master": true,
  "quantity": 32.7
}

```

#### Example response:

```

HTTP/1.0 201 CREATED
Content-Type: application/json

{
  "barcode": "SKT52680",
  "id": 1,
  "main": true,
  "master": true,
  "quantity": 32.7
}

```

**/api/items/<id>/barcodes/<id>**

**GET /storekeeper/api/items/ (int: item\_id) /barcodes/**  
**int: id** Get barcode

#### Parameters

- **id** – ID of selected barcode for get
- **item\_id** – ID of item

#### Status Codes

- 200 OK – no error
- 401 Unauthorized – user was not logged in
- 404 Not Found – there is no barcode

**Example request:**

```
GET /storekeeper/api/items/1/barcodes/1 HTTP/1.1
Host: localhost:8000
Content-Type: application/json
```

**Example response:**

```
HTTP/1.0 200 OK
Content-Type: application/json

{
  "barcode": "SKT52680",
  "id": 1,
  "main": true,
  "master": true,
  "quantity": 32.7
}
```

**PUT /storekeeper/api/items/ (int: item\_id) /barcodes/**  
**int: id** Update barcode

**Parameters**

- **id** – ID of selected barcode for put
- **item\_id** – ID of item

**Status Codes**

- **200 OK** – no error
- **401 Unauthorized** – user was not logged in
- **404 Not Found** – there is no barcode
- **422 Unprocessable Entity** – there is wrong type / missing field / can not add one barcode twice / can not set non-main barcode as master barcode / can not set more than one master barcode to an item

**Example request:**

```
PUT /storekeeper/api/items/1/barcodes/1 HTTP/1.1
Host: localhost:8000
Content-Type: application/json

{
  "barcode": "SKT52680",
  "master": true,
  "quantity": 32.7
}
```

**Example response:**

```
HTTP/1.0 200 OK
Content-Type: application/json

{
  "barcode": "SKT52680",
  "id": 1,
  "main": true,
  "master": true,
  "quantity": 32.7
}
```

**DELETE /storekeeper/api/items/ (int: item\_id) /barcodes/**  
**int: id** Delete barcode

**Parameters**

- **id** – ID of selected barcode for delete
- **item\_id** – ID of item

**Status Codes**

- 200 OK – no error
- 401 Unauthorized – user was not logged in
- 404 Not Found – there is no barcode

**Example request:**

```
DELETE /storekeeper/api/items/1/barcodes/1 HTTP/1.1
Host: localhost:8000
Content-Type: application/json
```

**Example response:**

```
HTTP/1.0 200 OK
Content-Type: application/json

null
```

**Commands**

**/api/items/<id>/barcodes/<id>/print**

**GET /storekeeper/api/items/ (int: item\_id) /barcodes/**  
**int: id/print** Generate barcode label to PDF with some details, and starts downloading that.

**Parameters**

- **id** – ID of selected barcode for get
- **item\_id** – ID of item

**Status Codes**

- 200 OK – no error
- 401 Unauthorized – user was not logged in
- 404 Not Found – there is no barcode

**Example request:**

```
GET /storekeeper/api/items/1/barcodes/1/print HTTP/1.1
Host: localhost:8000
Content-Type: application/json
```

**Example response:**

```
HTTP/1.0 200 OK
Content-Type: application/pdf
Content-Disposition: attachment; filename=label__SK642031__4f0ff51c73703295643a325e55bc7ed2d9
Content-Length: 11234

<file content>
```

**PUT /storekeeper/api/items/ (int: item\_id) /barcodes/**  
**int: id/print** Print barcode label with some details

**Parameters**

- **id** – ID of selected barcode for get

- `item_id` – ID of item

#### Status Codes

- 200 OK – no error
- 400 Bad Request – missing pycups python3 module
- 401 Unauthorized – user was not logged in
- 404 Not Found – there is no barcode
- 422 Unprocessable Entity – there is wrong type / missing field

#### Example request:

```
PUT /storekeeper/api/items/1/barcodes/1/print HTTP/1.1
Host: localhost:8000
Content-Type: application/json

{
  "copies": 3
}
```

#### Example response:

```
HTTP/1.0 200 OK
Content-Type: application/json

null
```

**Session** API endpoint for manage the current session.

#### Session management

##### /api/session

**GET** /storekeeper/api/session

Get current session

#### Status Codes

- 200 OK – no error
- 401 Unauthorized – user was not logged in

#### Example request:

```
GET /storekeeper/api/session HTTP/1.1
Host: localhost:8000
Content-Type: application/json
```

#### Example response:

```
HTTP/1.0 200 OK
Content-Type: application/json

{
  "admin": true,
  "disabled": false,
  "email": "admin@test.com",
  "id": 1,
  "username": "admin"
}
```

**POST** /storekeeper/api/session

Login user

### Status Codes

- 201 Created – no error
- 401 Unauthorized – bad authentication data or user is disabled
- 422 Unprocessable Entity – there is wrong type / missing field

#### Example request:

```
POST /storekeeper/api/session HTTP/1.1
Host: localhost:8000
Content-Type: application/json

{
  "password": "secret",
  "remember": false,
  "username": "admin"
}
```

#### Example response:

```
HTTP/1.0 201 CREATED
Content-Type: application/json

{
  "admin": true,
  "disabled": false,
  "email": "admin@test.com",
  "id": 1,
  "username": "admin"
}
```

### DELETE /storekeeper/api/session

Logout user

### Status Codes

- 200 OK – no error
- 401 Unauthorized – user was not logged in

#### Example request:

```
DELETE /storekeeper/api/session HTTP/1.1
Host: localhost:8000
Content-Type: application/json
```

#### Example response:

```
HTTP/1.0 200 OK
Content-Type: application/json

null
```

**Stocktakings** API endpoint for manage stocktaking results.

### Data management

#### /api/stocktakings

### GET /storekeeper/api/stocktakings

List stocktakings

### Status Codes

- 200 OK – no error
- 401 Unauthorized – user was not logged in

### Example request:

```
GET /storekeeper/api/stocktakings HTTP/1.1
Host: localhost:8000
Content-Type: application/json
```

### Example response:

```
HTTP/1.0 200 OK
Content-Type: application/json

[
  {
    "close_timestamp": null,
    "close_user": {
      "admin": null,
      "disabled": null,
      "email": null,
      "id": null,
      "username": null
    },
    "comment": "Maybe missing some items",
    "id": 1,
    "timestamp": "2016-03-18T07:58:11.557176+00:00"
  },
  {
    "close_timestamp": null,
    "close_user": {
      "admin": null,
      "disabled": null,
      "email": null,
      "id": null,
      "username": null
    },
    "comment": null,
    "id": 2,
    "timestamp": "2016-03-18T07:58:11.557202+00:00"
  }
]
```

## POST /storekeeper/api/stocktakings

Create stocktaking

### Status Codes

- 201 Created – no error
- 401 Unauthorized – user was not logged in
- 422 Unprocessable Entity – there is wrong type / missing field

### Example request:

```
POST /storekeeper/api/stocktakings HTTP/1.1
Host: localhost:8000
Content-Type: application/json

{
  "comment": "Maybe missing some items"
}
```

### Example response:

```

HTTP/1.0 201 CREATED
Content-Type: application/json

{
  "close_timestamp": null,
  "close_user": {
    "admin": null,
    "disabled": null,
    "email": null,
    "id": null,
    "username": null
  },
  "comment": "Maybe missing some items",
  "id": 1,
  "timestamp": "2016-03-18T07:58:11.557176+00:00"
}

```

**/api/stocktakings/<id>**

**GET /storekeeper/api/stocktakings/ (int: id)**

Get stocktaking

#### Parameters

- **id** – ID of selected stocktaking for get

#### Status Codes

- **200 OK** – no error
- **401 Unauthorized** – user was not logged in
- **404 Not Found** – there is no stocktaking

**Example request:**

```

GET /storekeeper/api/stocktakings/1 HTTP/1.1
Host: localhost:8000
Content-Type: application/json

```

**Example response:**

```

HTTP/1.0 200 OK
Content-Type: application/json

{
  "close_timestamp": null,
  "close_user": {
    "admin": null,
    "disabled": null,
    "email": null,
    "id": null,
    "username": null
  },
  "comment": "Maybe missing some items",
  "id": 1,
  "timestamp": "2016-03-18T07:58:11.557176+00:00"
}

```

**PUT /storekeeper/api/stocktakings/ (int: id)**

Update stocktaking

#### Parameters

- **id** – ID of selected stocktaking for put

#### Status Codes

- 200 OK – no error
- 401 Unauthorized – user was not logged in
- 404 Not Found – there is no stocktaking
- 422 Unprocessable Entity – there is wrong type / missing field

### Example request:

```
PUT /storekeeper/api/stocktakings/1 HTTP/1.1
Host: localhost:8000
Content-Type: application/json

{
  "comment": "A box has been damaged"
}
```

### Example response:

```
HTTP/1.0 200 OK
Content-Type: application/json

{
  "close_timestamp": null,
  "close_user": {
    "admin": null,
    "disabled": null,
    "email": null,
    "id": null,
    "username": null
  },
  "comment": "A box has been damaged",
  "id": 1,
  "timestamp": "2016-03-18T07:58:11.557176+00:00"
}
```

## DELETE /storekeeper/api/stocktakings/ (int: id)

Delete stocktaking

### Parameters

- **id** – ID of selected stocktaking for delete

### Status Codes

- 200 OK – no error
- 401 Unauthorized – user was not logged in
- 404 Not Found – there is no stocktaking

### Example request:

```
DELETE /storekeeper/api/stocktakings/1 HTTP/1.1
Host: localhost:8000
Content-Type: application/json
```

### Example response:

```
HTTP/1.0 200 OK
Content-Type: application/json

null
```

## /api/stocktakings/<id>/items



**GET** /storekeeper/api/stocktakings/ (int: *id*) /items

List stocktaking items.

#### Parameters

- **id** – ID of stocktaking

#### Status Codes

- 200 OK – no error
- 401 Unauthorized – user was not logged in
- 404 Not Found – there is no item

#### Example request:

```
GET /storekeeper/api/stocktakings/1/items HTTP/1.1
Host: localhost:8000
Content-Type: application/json
```

#### Example response:

```
HTTP/1.0 200 OK
Content-Type: application/json

[
  {
    "id": 1,
    "item": {
      "article_number": "FO213546",
      "id": 2,
      "location": "B02/01",
      "name": "Pipe",
      "purchase_price": 0.0,
      "quantity": 0.0,
      "unit": {
        "id": 1,
        "unit": "m"
      },
      "vendor": {
        "id": 2,
        "name": "Star Shop Ltd."
      },
      "warning_quantity": 0.0
    },
    "quantity": 52.1
  },
  {
    "id": 2,
    "item": {
      "article_number": "SK132465",
      "id": 1,
      "location": "A12/12",
      "name": "Spray",
      "purchase_price": 60.4,
      "quantity": 0.0,
      "unit": {
        "id": 2,
        "unit": "pcs"
      },
      "vendor": {
        "id": 1,
        "name": "Heavy Duty Ltd."
      },
      "warning_quantity": 4.0
    }
  }
]
```

```
    },  
    "quantity": 26.8  
  }  
]
```

**POST /storekeeper/api/stocktakings/ (int: id) /items**

Create stocktaking item

**Parameters**

- **id** – ID of stocktaking

**Status Codes**

- 201 Created – no error
- 401 Unauthorized – user was not logged in
- 403 Forbidden – can not add new stocktakings item after items was closed
- 404 Not Found – there is no item
- 422 Unprocessable Entity – there is wrong type / missing field / can not add one item twice

**Example request:**

```
POST /storekeeper/api/stocktakings/1/items HTTP/1.1  
Host: localhost:8000  
Content-Type: application/json  
  
{  
  "item": {  
    "article_number": "FO213546",  
    "id": 2,  
    "location": "B02/01",  
    "name": "Pipe",  
    "purchase_price": 0.0,  
    "quantity": 0.0,  
    "unit": {  
      "id": 1,  
      "unit": "m"  
    },  
    "vendor": {  
      "id": 2,  
      "name": "Star Shop Ltd."  
    },  
    "warning_quantity": 0.0  
  },  
  "quantity": 52.1  
}
```

**Example response:**

```
HTTP/1.0 201 CREATED  
Content-Type: application/json  
  
{  
  "id": 1,  
  "item": {  
    "article_number": "FO213546",  
    "id": 2,  
    "location": "B02/01",  
    "name": "Pipe",  
    "purchase_price": 0.0,  
    "quantity": 0.0,  
    "unit": {
```

```

    "id": 1,
    "unit": "m"
  },
  "vendor": {
    "id": 2,
    "name": "Star Shop Ltd."
  },
  "warning_quantity": 0.0
},
"quantity": 52.1
}

```

**/api/stocktakings/<id>/items/<item\_id>**

**GET /storekeeper/api/stocktakings/ (int: id) /items/**  
**int: item\_id** Get stocktaking item

#### Parameters

- **id** – ID of stocktaking
- **item\_id** – ID of selected stocktaking item for get

#### Status Codes

- **200 OK** – no error
- **401 Unauthorized** – user was not logged in
- **404 Not Found** – there is no stocktaking item

#### Example request:

```

GET /storekeeper/api/stocktakings/1/items/1 HTTP/1.1
Host: localhost:8000
Content-Type: application/json

```

#### Example response:

```

HTTP/1.0 200 OK
Content-Type: application/json

{
  "id": 1,
  "item": {
    "article_number": "FO213546",
    "id": 2,
    "location": "B02/01",
    "name": "Pipe",
    "purchase_price": 0.0,
    "quantity": 0.0,
    "unit": {
      "id": 1,
      "unit": "m"
    },
    "vendor": {
      "id": 2,
      "name": "Star Shop Ltd."
    },
    "warning_quantity": 0.0
  },
  "quantity": 52.1
}

```

**PUT /storekeeper/api/stocktakings/ (int: id) /items/**  
**int: item\_id** Update stocktaking item

### Parameters

- **id** – ID of stocktaking
- **item\_id** – ID of selected stocktaking item for put

### Status Codes

- 200 OK – no error
- 401 Unauthorized – user was not logged in
- 403 Forbidden – can not change work item after outbound/returned items was closed
- 404 Not Found – there is no stocktaking item
- 422 Unprocessable Entity – there is wrong type / missing field / can not add one item twice

### Example request:

```
PUT /storekeeper/api/stocktakings/1/items/1 HTTP/1.1
Host: localhost:8000
Content-Type: application/json

{
  "item": {
    "article_number": "FO213546",
    "id": 2,
    "location": "B02/01",
    "name": "Pipe",
    "purchase_price": 0.0,
    "quantity": 0.0,
    "unit": {
      "id": 1,
      "unit": "m"
    },
  },
  "vendor": {
    "id": 2,
    "name": "Star Shop Ltd."
  },
  "warning_quantity": 0.0
},
"quantity": 52.1
}
```

### Example response:

```
HTTP/1.0 200 OK
Content-Type: application/json

{
  "id": 1,
  "item": {
    "article_number": "FO213546",
    "id": 2,
    "location": "B02/01",
    "name": "Pipe",
    "purchase_price": 0.0,
    "quantity": 0.0,
    "unit": {
      "id": 1,
      "unit": "m"
    },
  },
  "vendor": {
    "id": 2,
    "name": "Star Shop Ltd."
  }
}
```

```

    },
    "warning_quantity": 0.0
  },
  "quantity": 52.1
}

```

**DELETE** `/storekeeper/api/stocktakings/ (int: id) /items/`  
**int: item\_id** Delete stocktaking item

#### Parameters

- **id** – ID of stocktaking
- **item\_id** – ID of selected stocktaking item for delete

#### Status Codes

- 200 OK – no error
- 401 Unauthorized – user was not logged in
- 403 Forbidden – can not delete stocktaking item after items was closed
- 404 Not Found – there is no stocktaking item

#### Example request:

```

DELETE /storekeeper/api/stocktakings/1/items/1 HTTP/1.1
Host: localhost:8000
Content-Type: application/json

```

#### Example response:

```

HTTP/1.0 200 OK
Content-Type: application/json

null

```

## Commands

`/api/stocktakings/<id>/close`  
**PUT** `/storekeeper/api/stocktakings/ (int: id) /close`  
 Close items on stocktaking

#### Parameters

- **id** – ID of selected stocktaking for put

#### Status Codes

- 200 OK – no error
- 401 Unauthorized – user was not logged in
- 404 Not Found – there is no stocktaking
- 422 Unprocessable Entity – there is wrong type / missing field / items have been closed / insufficient quantities for close the stocktaking items

#### Example request:

```

PUT /storekeeper/api/stocktakings/1/close HTTP/1.1
Host: localhost:8000
Content-Type: application/json

```

#### Example response:

```
HTTP/1.0 200 OK
Content-Type: application/json

{
  "close_timestamp": "2016-03-18T07:58:11.557219+00:00",
  "close_user": {
    "admin": false,
    "disabled": false,
    "email": "foo@bar.com",
    "id": 2,
    "username": "foo"
  },
  "comment": "Maybe missing some items",
  "id": 1,
  "timestamp": "2016-03-18T07:58:11.557176+00:00"
}
```

**Units** API endpoint for manage units.

## Data management

### /api/units

**GET /storekeeper/api/units**

List units

#### Status Codes

- 200 OK – no error
- 401 Unauthorized – user was not logged in

**Example request:**

```
GET /storekeeper/api/units HTTP/1.1
Host: localhost:8000
Content-Type: application/json
```

**Example response:**

```
HTTP/1.0 200 OK
Content-Type: application/json

[
  {
    "id": 1,
    "unit": "m"
  },
  {
    "id": 2,
    "unit": "pcs"
  }
]
```

**POST /storekeeper/api/units**

Create unit

#### Status Codes

- 201 Created – no error
- 401 Unauthorized – user was not logged in
- 422 Unprocessable Entity – there is wrong type / missing field / unit is already exist

**Example request:**

```
POST /storekeeper/api/units HTTP/1.1
Host: localhost:8000
Content-Type: application/json

{
  "unit": "m"
}
```

**Example response:**

```
HTTP/1.0 201 CREATED
Content-Type: application/json

{
  "id": 1,
  "unit": "m"
}
```

**/api/units/<id>****GET /storekeeper/api/units/ (int: id)**

Get unit

**Parameters**

- **id** – ID of selected unit for get

**Status Codes**

- 200 OK – no error
- 401 Unauthorized – user was not logged in
- 404 Not Found – there is no unit

**Example request:**

```
GET /storekeeper/api/units/1 HTTP/1.1
Host: localhost:8000
Content-Type: application/json
```

**Example response:**

```
HTTP/1.0 200 OK
Content-Type: application/json

{
  "id": 1,
  "unit": "m"
}
```

**PUT /storekeeper/api/units/ (int: id)**

Update unit

**Parameters**

- **id** – ID of selected unit for put

**Status Codes**

- 200 OK – no error
- 401 Unauthorized – user was not logged in
- 404 Not Found – there is no unit
- 422 Unprocessable Entity – there is wrong type / missing field / unit is already exist

**Example request:**

```
PUT /storekeeper/api/units/1 HTTP/1.1
Host: localhost:8000
Content-Type: application/json

{
  "unit": "dl"
}
```

**Example response:**

```
HTTP/1.0 200 OK
Content-Type: application/json

{
  "id": 1,
  "unit": "dl"
}
```

**DELETE /storekeeper/api/units/ (int: id)**

Delete unit

**Parameters**

- **id** – ID of selected unit for delete

**Status Codes**

- 200 OK – no error
- 401 Unauthorized – user was not logged in
- 404 Not Found – there is no unit

**Example request:**

```
DELETE /storekeeper/api/units/1 HTTP/1.1
Host: localhost:8000
Content-Type: application/json
```

**Example response:**

```
HTTP/1.0 200 OK
Content-Type: application/json

null
```

**Users** API endpoint for manage users.

**Data management**

**/api/users**

**GET /storekeeper/api/users**

List users (*for administrators only*)

**Status Codes**

- 200 OK – no error
- 401 Unauthorized – user was not logged in
- 403 Forbidden – user has not enough rights

**Example request:**



```
GET /storekeeper/api/users HTTP/1.1
Host: localhost:8000
Content-Type: application/json
```

**Example response:**

```
HTTP/1.0 200 OK
Content-Type: application/json

[
  {
    "admin": true,
    "disabled": false,
    "email": "admin@test.com",
    "id": 1,
    "username": "admin"
  },
  {
    "admin": false,
    "disabled": false,
    "email": "foo@bar.com",
    "id": 2,
    "username": "foo"
  }
]
```

**POST /storekeeper/api/users**

Create user (*for administrators only*)

**Status Codes**

- 201 Created – no error
- 401 Unauthorized – user was not logged in
- 403 Forbidden – user has not enough rights
- 422 Unprocessable Entity – {original} / user is already exist

**Example request:**

```
POST /storekeeper/api/users HTTP/1.1
Host: localhost:8000
Content-Type: application/json

{
  "email": "foo@bar.com",
  "password": "bar",
  "username": "foo"
}
```

**Example response:**

```
HTTP/1.0 201 CREATED
Content-Type: application/json

{
  "admin": false,
  "disabled": false,
  "email": "foo@bar.com",
  "id": 2,
  "username": "foo"
}
```

**/api/users/<id>**

**GET** /storekeeper/api/users/ (int: id)

Get user

**Parameters**

- **id** – ID of selected user for get

**Status Codes**

- 200 OK – no error
- 401 Unauthorized – user was not logged in
- 404 Not Found – there is no user

**Example request:**

```
GET /storekeeper/api/users/2 HTTP/1.1
Host: localhost:8000
Content-Type: application/json
```

**Example response:**

```
HTTP/1.0 200 OK
Content-Type: application/json

{
  "admin": false,
  "disabled": false,
  "email": "foo@bar.com",
  "id": 2,
  "username": "foo"
}
```

**PUT** /storekeeper/api/users/ (int: id)

Update user

**Parameters**

- **id** – ID of selected user for put

**Status Codes**

- 200 OK – no error
- 401 Unauthorized – user was not logged in
- 403 Forbidden – user can not modify other users
- 404 Not Found – there is no user
- 422 Unprocessable Entity – {original} / user is already exist

**Example request:**

```
PUT /storekeeper/api/users/2 HTTP/1.1
Host: localhost:8000
Content-Type: application/json

{
  "email": "foo@bar.com",
  "password": "bar",
  "username": "new_foo"
}
```

**Example response:**

```
HTTP/1.0 200 OK
Content-Type: application/json
```

```
{
  "admin": false,
  "disabled": false,
  "email": "foo@bar.com",
  "id": 2,
  "username": "new_foo"
}
```

**DELETE /storekeeper/api/users/ (int: id)**Delete user (*for administrators only*)**Parameters**

- **id** – ID of selected user for delete

**Status Codes**

- 200 OK – no error
- 401 Unauthorized – user was not logged in
- 403 Forbidden – user can not remove itself
- 404 Not Found – there is no user

**Example request:**

```
DELETE /storekeeper/api/users/2 HTTP/1.1
Host: localhost:8000
Content-Type: application/json
```

**Example response:**

```
HTTP/1.0 200 OK
Content-Type: application/json

null
```

**Config management****/api/users/<id>/config****GET /storekeeper/api/users/ (int: id) /config**

List user items.

**Parameters**

- **id** – ID of user

**Status Codes**

- 200 OK – no error
- 401 Unauthorized – user was not logged in
- 404 Not Found – there is no item

**Example request:**

```
GET /storekeeper/api/users/2/config HTTP/1.1
Host: localhost:8000
Content-Type: application/json
```

**Example response:**

```
HTTP/1.0 200 OK
Content-Type: application/json

[
  {
    "name": "lang",
    "value": "hu"
  },
  {
    "name": "fruits",
    "value": "[\"apple\", \"orange\", \"banana\"]"
  }
]
```

#### **POST /storekeeper/api/users/ (int: id) /config**

Create user item

##### **Parameters**

- **id** – ID of user

##### **Status Codes**

- 201 Created – no error
- 401 Unauthorized – user was not logged in
- 404 Not Found – there is no item
- 422 Unprocessable Entity – there is wrong type / missing field / can not add one item twice

##### **Example request:**

```
POST /storekeeper/api/users/2/config HTTP/1.1
Host: localhost:8000
Content-Type: application/json

{
  "name": "lang",
  "value": "hu"
}
```

##### **Example response:**

```
HTTP/1.0 201 CREATED
Content-Type: application/json

{
  "name": "lang",
  "value": "hu"
}
```

#### **/api/users/<id>/config/<id>**

#### **GET /storekeeper/api/users/ (int: id) /config/**

**string: name** Get user item

##### **Parameters**

- **id** – ID of user
- **name** – Name of selected user config value for get

##### **Status Codes**

- 200 OK – no error
- 401 Unauthorized – user was not logged in

- 404 Not Found – there is no user item

**Example request:**

```
GET /storekeeper/api/users/2/config/lang HTTP/1.1
Host: localhost:8000
Content-Type: application/json
```

**Example response:**

```
HTTP/1.0 200 OK
Content-Type: application/json

{
  "name": "lang",
  "value": "hu"
}
```

**PUT /storekeeper/api/users/ (int: id) /config/**  
**string: name** Update user item

**Parameters**

- **id** – ID of user
- **name** – Name of selected user config value for put

**Status Codes**

- 200 OK – no error
- 401 Unauthorized – user was not logged in
- 404 Not Found – there is no user item
- 422 Unprocessable Entity – there is wrong type / missing field / can not use one config name twice

**Example request:**

```
PUT /storekeeper/api/users/2/config/lang HTTP/1.1
Host: localhost:8000
Content-Type: application/json

{
  "name": "lang",
  "value": "hu"
}
```

**Example response:**

```
HTTP/1.0 200 OK
Content-Type: application/json

{
  "name": "lang",
  "value": "hu"
}
```

**DELETE /storekeeper/api/users/ (int: id) /config/**  
**string: name** Delete user item

**Parameters**

- **id** – ID of user
- **name** – Name of selected user config value for delete

**Status Codes**

- 200 OK – no error
- 401 Unauthorized – user was not logged in
- 404 Not Found – there is no user item

### Example request:

```
DELETE /storekeeper/api/users/2/config/lang HTTP/1.1
Host: localhost:8000
Content-Type: application/json
```

### Example response:

```
HTTP/1.0 200 OK
Content-Type: application/json

null
```

**Vendors** API endpoint for manage vendors.

## Data management

### /api/vendors

**GET /storekeeper/api/vendors**

List vendors

#### Status Codes

- 200 OK – no error
- 401 Unauthorized – user was not logged in

### Example request:

```
GET /storekeeper/api/vendors HTTP/1.1
Host: localhost:8000
Content-Type: application/json
```

### Example response:

```
HTTP/1.0 200 OK
Content-Type: application/json

[
  {
    "id": 1,
    "name": "Heavy Duty Ltd."
  },
  {
    "id": 2,
    "name": "Star Shop Ltd."
  }
]
```

**POST /storekeeper/api/vendors**

Create vendor

#### Status Codes

- 201 Created – no error
- 401 Unauthorized – user was not logged in
- 422 Unprocessable Entity – {original} / vendor is already exist

**Example request:**

```
POST /storekeeper/api/vendors HTTP/1.1
Host: localhost:8000
Content-Type: application/json

{
  "name": "Heavy Duty Ltd."
}
```

**Example response:**

```
HTTP/1.0 201 CREATED
Content-Type: application/json

{
  "id": 1,
  "name": "Heavy Duty Ltd."
}
```

**/api/vendors/<id>****GET /storekeeper/api/vendors/ (int: id)**

Get vendor

**Parameters**

- **id** – ID of selected vendor for get

**Status Codes**

- 200 OK – no error
- 401 Unauthorized – user was not logged in
- 404 Not Found – there is no vendor

**Example request:**

```
GET /storekeeper/api/vendors/1 HTTP/1.1
Host: localhost:8000
Content-Type: application/json
```

**Example response:**

```
HTTP/1.0 200 OK
Content-Type: application/json

{
  "id": 1,
  "name": "Heavy Duty Ltd."
}
```

**PUT /storekeeper/api/vendors/ (int: id)**

Update vendor

**Parameters**

- **id** – ID of selected vendor for put

**Status Codes**

- 200 OK – no error
- 401 Unauthorized – user was not logged in
- 404 Not Found – there is no vendor
- 422 Unprocessable Entity – {original} / vendor is already exist

**Example request:**

```
PUT /storekeeper/api/vendors/1 HTTP/1.1
Host: localhost:8000
Content-Type: application/json

{
  "name": "new_foo"
}
```

**Example response:**

```
HTTP/1.0 200 OK
Content-Type: application/json

{
  "id": 1,
  "name": "new_foo"
}
```

**DELETE /storekeeper/api/vendors/ (int: id)**

Delete vendor

**Parameters**

- **id** – ID of selected vendor for delete

**Status Codes**

- 200 OK – no error
- 401 Unauthorized – user was not logged in
- 404 Not Found – there is no vendor

**Example request:**

```
DELETE /storekeeper/api/vendors/1 HTTP/1.1
Host: localhost:8000
Content-Type: application/json
```

**Example response:**

```
HTTP/1.0 200 OK
Content-Type: application/json

null
```

**Works** API endpoint for manage works.

**Data management**

**/api/works**

**GET /storekeeper/api/works**

List works

**Status Codes**

- 200 OK – no error
- 401 Unauthorized – user was not logged in

**Example request:**



```
GET /storekeeper/api/works HTTP/1.1
Host: localhost:8000
Content-Type: application/json
```

**Example response:**

```
HTTP/1.0 200 OK
Content-Type: application/json

[
  {
    "comment": "First work",
    "customer": {
      "id": 1,
      "name": "Big Customer Ltd."
    },
    "id": 1,
    "outbound_close_timestamp": null,
    "outbound_close_user": {
      "admin": null,
      "disabled": null,
      "email": null,
      "id": null,
      "username": null
    },
    "returned_close_timestamp": null,
    "returned_close_user": {
      "admin": null,
      "disabled": null,
      "email": null,
      "id": null,
      "username": null
    }
  },
  {
    "comment": null,
    "customer": {
      "id": 2,
      "name": "Buy Everything Co."
    },
    "id": 2,
    "outbound_close_timestamp": null,
    "outbound_close_user": {
      "admin": null,
      "disabled": null,
      "email": null,
      "id": null,
      "username": null
    },
    "returned_close_timestamp": null,
    "returned_close_user": {
      "admin": null,
      "disabled": null,
      "email": null,
      "id": null,
      "username": null
    }
  }
]
```

**POST /storekeeper/api/works**  
Create work

**Status Codes**

- 201 Created – no error
- 401 Unauthorized – user was not logged in
- 422 Unprocessable Entity – there is wrong type / missing field

### Example request:

```
POST /storekeeper/api/works HTTP/1.1
Host: localhost:8000
Content-Type: application/json

{
  "comment": "First work",
  "customer": {
    "id": 1,
    "name": "Big Customer Ltd."
  }
}
```

### Example response:

```
HTTP/1.0 201 CREATED
Content-Type: application/json

{
  "comment": "First work",
  "customer": {
    "id": 1,
    "name": "Big Customer Ltd."
  },
  "id": 1,
  "outbound_close_timestamp": null,
  "outbound_close_user": {
    "admin": null,
    "disabled": null,
    "email": null,
    "id": null,
    "username": null
  },
  "returned_close_timestamp": null,
  "returned_close_user": {
    "admin": null,
    "disabled": null,
    "email": null,
    "id": null,
    "username": null
  }
}
```

/api/works/<id>

GET /storekeeper/api/works/ (int: id)

Get work

### Parameters

- **id** – ID of selected work for get

### Status Codes

- 200 OK – no error
- 401 Unauthorized – user was not logged in
- 404 Not Found – there is no work

**Example request:**

```
GET /storekeeper/api/works/1 HTTP/1.1
Host: localhost:8000
Content-Type: application/json
```

**Example response:**

```
HTTP/1.0 200 OK
Content-Type: application/json

{
  "comment": "First work",
  "customer": {
    "id": 1,
    "name": "Big Customer Ltd."
  },
  "id": 1,
  "outbound_close_timestamp": null,
  "outbound_close_user": {
    "admin": null,
    "disabled": null,
    "email": null,
    "id": null,
    "username": null
  },
  "returned_close_timestamp": null,
  "returned_close_user": {
    "admin": null,
    "disabled": null,
    "email": null,
    "id": null,
    "username": null
  }
}
```

**PUT** /storekeeper/api/works/ (int: *id*)  
Update work

**Parameters**

- **id** – ID of selected work for put

**Status Codes**

- 200 OK – no error
- 401 Unauthorized – user was not logged in
- 404 Not Found – there is no work
- 422 Unprocessable Entity – there is wrong type / missing field

**Example request:**

```
PUT /storekeeper/api/works/1 HTTP/1.1
Host: localhost:8000
Content-Type: application/json

{
  "comment": "Something are not finished",
  "customer": {
    "id": 1,
    "name": "Big Customer Ltd."
  }
}
```

**Example response:**

```
HTTP/1.0 200 OK
Content-Type: application/json

{
  "comment": "Something are not finished",
  "customer": {
    "id": 1,
    "name": "Big Customer Ltd."
  },
  "id": 1,
  "outbound_close_timestamp": null,
  "outbound_close_user": {
    "admin": null,
    "disabled": null,
    "email": null,
    "id": null,
    "username": null
  },
  "returned_close_timestamp": null,
  "returned_close_user": {
    "admin": null,
    "disabled": null,
    "email": null,
    "id": null,
    "username": null
  }
}
```

**DELETE** /storekeeper/api/works/ (int: *id*)

Delete work

#### Parameters

- **id** – ID of selected work for delete

#### Status Codes

- 200 OK – no error
- 401 Unauthorized – user was not logged in
- 404 Not Found – there is no work

Example request:

```
DELETE /storekeeper/api/works/1 HTTP/1.1
Host: localhost:8000
Content-Type: application/json
```

Example response:

```
HTTP/1.0 200 OK
Content-Type: application/json

null
```

/api/works/<id>/items

**GET** /storekeeper/api/works/ (int: *id*) /items

List work items

#### Parameters

- **id** – ID of work

#### Status Codes

- 200 OK – no error
- 401 Unauthorized – user was not logged in
- 404 Not Found – there is no item

**Example request:**

```
GET /storekeeper/api/works/1/items HTTP/1.1
Host: localhost:8000
Content-Type: application/json
```

**Example response:**

```
HTTP/1.0 200 OK
Content-Type: application/json

[
  {
    "id": 1,
    "item": {
      "article_number": "FO213546",
      "id": 2,
      "location": "B02/01",
      "name": "Pipe",
      "purchase_price": 0.0,
      "quantity": 0.0,
      "unit": {
        "id": 1,
        "unit": "m"
      },
      "vendor": {
        "id": 2,
        "name": "Star Shop Ltd."
      },
      "warning_quantity": 0.0
    },
    "outbound_quantity": 132.8,
    "returned_quantity": null
  },
  {
    "id": 2,
    "item": {
      "article_number": "SK132465",
      "id": 1,
      "location": "A12/12",
      "name": "Spray",
      "purchase_price": 60.4,
      "quantity": 0.0,
      "unit": {
        "id": 2,
        "unit": "pcs"
      },
      "vendor": {
        "id": 1,
        "name": "Heavy Duty Ltd."
      },
      "warning_quantity": 4.0
    },
    "outbound_quantity": 41.2,
    "returned_quantity": 2.1
  }
]
```

**POST /storekeeper/api/works/ (int: id) /items**  
Create work item

### Parameters

- **id** – ID of work

### Status Codes

- 201 Created – no error
- 401 Unauthorized – user was not logged in
- 403 Forbidden – can not add new work item after outbound items was closed
- 404 Not Found – there is no item
- 422 Unprocessable Entity – there is wrong type / missing field / can not add one item twice

### Example request:

```
POST /storekeeper/api/works/1/items HTTP/1.1
Host: localhost:8000
Content-Type: application/json

{
  "item": {
    "article_number": "FO213546",
    "id": 2,
    "location": "B02/01",
    "name": "Pipe",
    "purchase_price": 0.0,
    "quantity": 0.0,
    "unit": {
      "id": 1,
      "unit": "m"
    },
  },
  "vendor": {
    "id": 2,
    "name": "Star Shop Ltd."
  },
  "warning_quantity": 0.0
},
"outbound_quantity": 132.8
}
```

### Example response:

```
HTTP/1.0 201 CREATED
Content-Type: application/json

{
  "id": 1,
  "item": {
    "article_number": "FO213546",
    "id": 2,
    "location": "B02/01",
    "name": "Pipe",
    "purchase_price": 0.0,
    "quantity": 0.0,
    "unit": {
      "id": 1,
      "unit": "m"
    },
  },
  "vendor": {
    "id": 2,
    "name": "Star Shop Ltd."
  },
}
```

```

    "warning_quantity": 0.0
  },
  "outbound_quantity": 132.8,
  "returned_quantity": null
}

```

**/api/works/<id>/items/<item\_id>**

**GET** /storekeeper/api/works/ (int: id) /items/  
int: item\_id Get work item

#### Parameters

- **id** – ID of work
- **item\_id** – ID of selected work item for get

#### Status Codes

- 200 OK – no error
- 401 Unauthorized – user was not logged in
- 404 Not Found – there is no work item

#### Example request:

```

GET /storekeeper/api/works/1/items/1 HTTP/1.1
Host: localhost:8000
Content-Type: application/json

```

#### Example response:

```

HTTP/1.0 200 OK
Content-Type: application/json

{
  "id": 1,
  "item": {
    "article_number": "FO213546",
    "id": 2,
    "location": "B02/01",
    "name": "Pipe",
    "purchase_price": 0.0,
    "quantity": 0.0,
    "unit": {
      "id": 1,
      "unit": "m"
    },
    "vendor": {
      "id": 2,
      "name": "Star Shop Ltd."
    },
    "warning_quantity": 0.0
  },
  "outbound_quantity": 132.8,
  "returned_quantity": null
}

```

**PUT** /storekeeper/api/works/ (int: id) /items/  
int: item\_id Update work item

#### Parameters

- **id** – ID of work
- **item\_id** – ID of selected work item for get

### Status Codes

- 200 OK – no error
- 401 Unauthorized – user was not logged in
- 403 Forbidden – can not change work item after outbound/returned items was closed
- 404 Not Found – there is no work item
- 422 Unprocessable Entity – there is wrong type / missing field / can not add one item twice

### Example request:

```
PUT /storekeeper/api/works/1/items/1 HTTP/1.1
Host: localhost:8000
Content-Type: application/json

{
  "item": {
    "article_number": "FO213546",
    "id": 2,
    "location": "B02/01",
    "name": "Pipe",
    "purchase_price": 0.0,
    "quantity": 0.0,
    "unit": {
      "id": 1,
      "unit": "m"
    },
  },
  "vendor": {
    "id": 2,
    "name": "Star Shop Ltd."
  },
  "warning_quantity": 0.0
},
"outbound_quantity": 132.8
}
```

### Example response:

```
HTTP/1.0 200 OK
Content-Type: application/json

{
  "id": 1,
  "item": {
    "article_number": "FO213546",
    "id": 2,
    "location": "B02/01",
    "name": "Pipe",
    "purchase_price": 0.0,
    "quantity": 0.0,
    "unit": {
      "id": 1,
      "unit": "m"
    },
  },
  "vendor": {
    "id": 2,
    "name": "Star Shop Ltd."
  },
  "warning_quantity": 0.0
},
"outbound_quantity": 132.8,
```



```
"returned_quantity": null
}
```

**DELETE** /storekeeper/api/works/ (int: id) /items/  
int: item\_id Delete work item

#### Parameters

- **id** – ID of work
- **item\_id** – ID of selected work item for get

#### Status Codes

- 200 OK – no error
- 401 Unauthorized – user was not logged in
- 403 Forbidden – can not delete work item after outbound/returned items was closed
- 404 Not Found – there is no work item

#### Example request:

```
DELETE /storekeeper/api/works/1/items/1 HTTP/1.1
Host: localhost:8000
Content-Type: application/json
```

#### Example response:

```
HTTP/1.0 200 OK
Content-Type: application/json

null
```

## Commands

/api/works/<id>/close-outbound  
**PUT** /storekeeper/api/works/ (int: id) /close-outbound  
Close outbound items on work

#### Parameters

- **id** – ID of selected work for put

#### Status Codes

- 200 OK – no error
- 401 Unauthorized – user was not logged in
- 404 Not Found – there is no work
- 422 Unprocessable Entity – there is wrong type / missing field / outbound items have been closed /insufficient quantities for close the outbound work items

#### Example request:

```
PUT /storekeeper/api/works/1/close-outbound HTTP/1.1
Host: localhost:8000
Content-Type: application/json
```

#### Example response:

```
HTTP/1.0 200 OK
Content-Type: application/json

{
```

```
"comment": "First work",
"customer": {
  "id": 1,
  "name": "Big Customer Ltd."
},
"id": 1,
"outbound_close_timestamp": "2016-03-18T07:58:11.557633+00:00",
"outbound_close_user": {
  "admin": false,
  "disabled": false,
  "email": "foo@bar.com",
  "id": 2,
  "username": "foo"
},
"returned_close_timestamp": null,
"returned_close_user": {
  "admin": null,
  "disabled": null,
  "email": null,
  "id": null,
  "username": null
}
}
```

**/api/works/<id>/close-returned**

**PUT /storekeeper/api/works/ (int: id) /close-returned**

Close returned items on work

#### Parameters

- **id** – ID of selected work for put

#### Status Codes

- 200 OK – no error
- 401 Unauthorized – user was not logged in
- 404 Not Found – there is no work
- 422 Unprocessable Entity – there is wrong type / missing field / outbound items have not been closed / returned items have been closed /insufficient quantities for close the returned work items

#### Example request:

```
PUT /storekeeper/api/works/1/close-returned HTTP/1.1
Host: localhost:8000
Content-Type: application/json
```

#### Example response:

```
HTTP/1.0 200 OK
Content-Type: application/json

{
  "comment": "First work",
  "customer": {
    "id": 1,
    "name": "Big Customer Ltd."
  },
  "id": 1,
  "outbound_close_timestamp": "2016-03-18T07:58:11.557633+00:00",
  "outbound_close_user": {
    "admin": false,
```

```

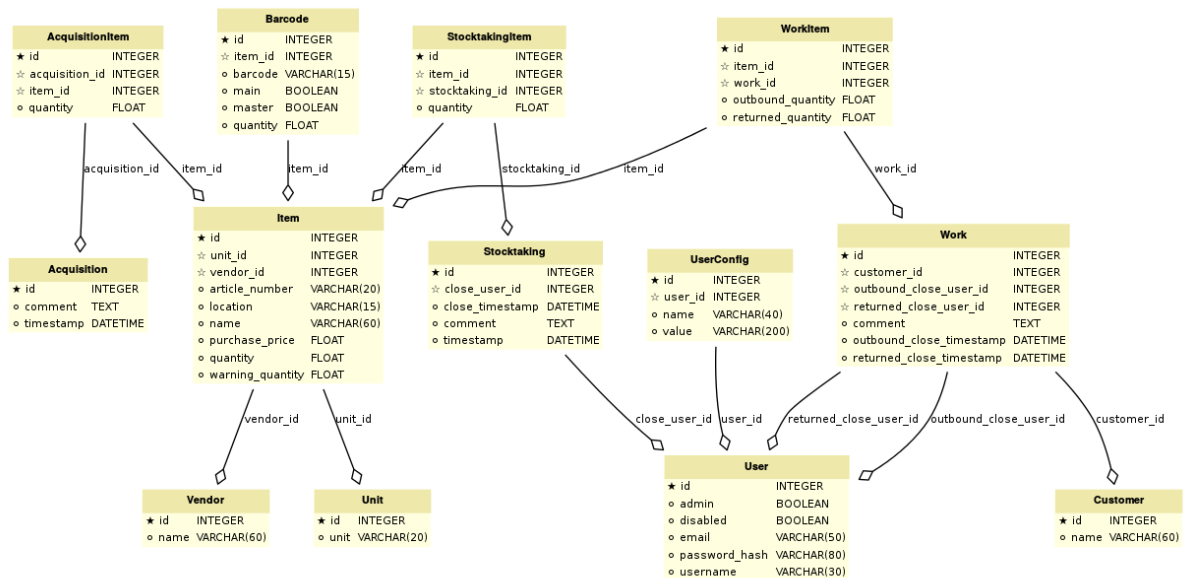
    "disabled": false,
    "email": "foo@bar.com",
    "id": 2,
    "username": "foo"
  },
  "returned_close_timestamp": "2016-03-18T07:58:11.557666+00:00",
  "returned_close_user": {
    "admin": false,
    "disabled": false,
    "email": "foo@bar.com",
    "id": 2,
    "username": "foo"
  }
}

```

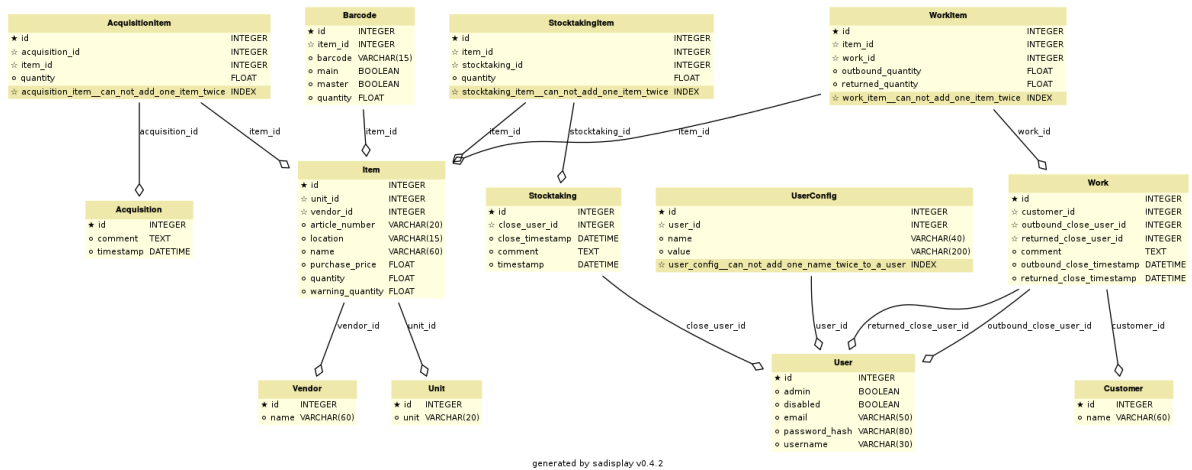
## 4.4.2 Details

### SQL Model

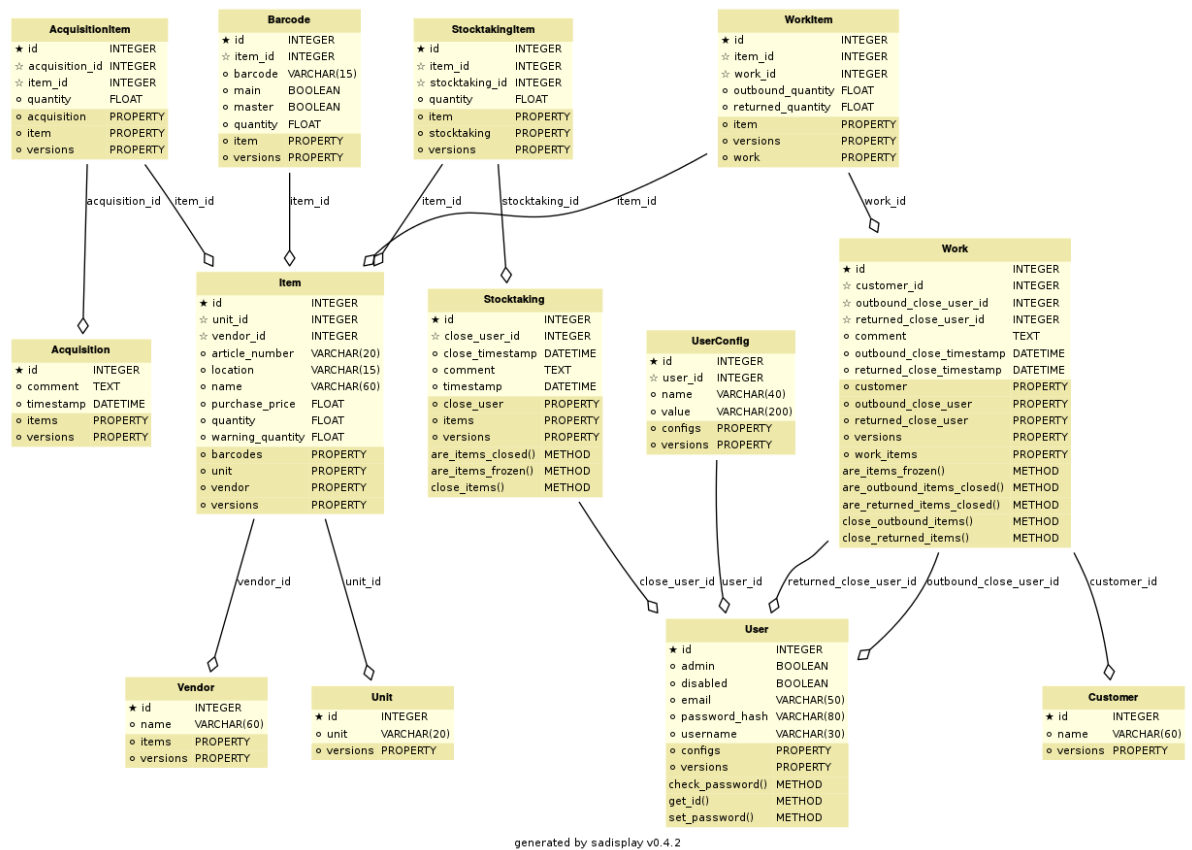
This software is SQL independent (thank you for [SQLAlchemy](#)), and we want to hold it in future. However, requirements, installer scripts, tests were prepared for and ran on PostgreSQL, MySQL and SQLite databases.



## Model with indexes



## Model with active properties and methods



---

## Indices and tables

---

- `genindex`



## /storekeeper

	GET /storekeeper/api/works, <a href="#">52</a>
GET /storekeeper/api/acquisitions, <a href="#">12</a>	GET /storekeeper/api/works/(int:id), <a href="#">54</a>
GET /storekeeper/api/acquisitions/(int:id), <a href="#">13</a>	GET /storekeeper/api/works/(int:id)/items, <a href="#">56</a>
GET /storekeeper/api/acquisitions/(int:id)/items, <a href="#">15</a>	GET /storekeeper/api/works/(int:id)/items/(int:it
GET /storekeeper/api/acquisitions/(int:id)/items/(int:item_id), <a href="#">17</a>	POST /storekeeper/api/acquisitions, <a href="#">13</a>
GET /storekeeper/api/barcodes, <a href="#">19</a>	POST /storekeeper/api/acquisitions/(int:id)/items, <a href="#">16</a>
GET /storekeeper/api/config, <a href="#">20</a>	POST /storekeeper/api/customers, <a href="#">21</a>
GET /storekeeper/api/customers, <a href="#">21</a>	POST /storekeeper/api/error, <a href="#">23</a>
GET /storekeeper/api/customers/(int:id), <a href="#">22</a>	POST /storekeeper/api/items, <a href="#">24</a>
GET /storekeeper/api/items, <a href="#">24</a>	POST /storekeeper/api/items/(int:id)/barcodes, <a href="#">29</a>
GET /storekeeper/api/items/(int:id), <a href="#">26</a>	POST /storekeeper/api/session, <a href="#">32</a>
GET /storekeeper/api/items/(int:id)/barcodes, <a href="#">28</a>	POST /storekeeper/api/stocktakings, <a href="#">34</a>
GET /storekeeper/api/items/(int:item_id)/barcodes/(int:id), <a href="#">29</a>	POST /storekeeper/api/stocktakings/(int:id)/items, <a href="#">38</a>
GET /storekeeper/api/items/(int:item_id)/barcodes/(int:id)/print, <a href="#">31</a>	POST /storekeeper/api/units, <a href="#">42</a>
GET /storekeeper/api/items/search, <a href="#">25</a>	POST /storekeeper/api/users, <a href="#">45</a>
GET /storekeeper/api/session, <a href="#">32</a>	POST /storekeeper/api/users/(int:id)/config, <a href="#">48</a>
GET /storekeeper/api/stocktakings, <a href="#">33</a>	POST /storekeeper/api/vendors, <a href="#">50</a>
GET /storekeeper/api/stocktakings/(int:id), <a href="#">35</a>	POST /storekeeper/api/works, <a href="#">53</a>
GET /storekeeper/api/stocktakings/(int:id)/items, <a href="#">36</a>	POST /storekeeper/api/works/(int:id)/items, <a href="#">57</a>
GET /storekeeper/api/stocktakings/(int:id)/items/(int:item_id), <a href="#">39</a>	PUT /storekeeper/api/acquisitions/(int:id), <a href="#">14</a>
GET /storekeeper/api/units, <a href="#">42</a>	PUT /storekeeper/api/acquisitions/(int:id)/items/, <a href="#">18</a>
GET /storekeeper/api/units/(int:id), <a href="#">43</a>	PUT /storekeeper/api/customers/(int:id), <a href="#">22</a>
GET /storekeeper/api/users, <a href="#">44</a>	PUT /storekeeper/api/items/(int:id), <a href="#">27</a>
GET /storekeeper/api/users/(int:id), <a href="#">45</a>	PUT /storekeeper/api/items/(int:item_id)/barcodes, <a href="#">30</a>
GET /storekeeper/api/users/(int:id)/config, <a href="#">47</a>	PUT /storekeeper/api/items/(int:item_id)/barcodes, <a href="#">31</a>
GET /storekeeper/api/users/(int:id)/config/(string:name), <a href="#">48</a>	PUT /storekeeper/api/stocktakings/(int:id), <a href="#">35</a>
GET /storekeeper/api/vendors, <a href="#">50</a>	PUT /storekeeper/api/stocktakings/(int:id)/close, <a href="#">41</a>
GET /storekeeper/api/vendors/(int:id), <a href="#">51</a>	PUT /storekeeper/api/stocktakings/(int:id)/items/, <a href="#">39</a>

---

```
PUT /storekeeper/api/units/(int:id),
    43
PUT /storekeeper/api/users/(int:id),
    46
PUT /storekeeper/api/users/(int:id)/config/(string:name),
    49
PUT /storekeeper/api/vendors/(int:id),
    51
PUT /storekeeper/api/works/(int:id),
    55
PUT /storekeeper/api/works/(int:id)/close-outbound,
    61
PUT /storekeeper/api/works/(int:id)/close-returned,
    62
PUT /storekeeper/api/works/(int:id)/items/(int:item_id),
    59
DELETE /storekeeper/api/acquisitions/(int:id),
    14
DELETE /storekeeper/api/acquisitions/(int:id)/items/(int:item_id),
    19
DELETE /storekeeper/api/customers/(int:id),
    22
DELETE /storekeeper/api/items/(int:id),
    28
DELETE /storekeeper/api/items/(int:item_id)/barcodes/(int:id),
    30
DELETE /storekeeper/api/session, 33
DELETE /storekeeper/api/stocktakings/(int:id),
    36
DELETE /storekeeper/api/stocktakings/(int:id)/items/(int:item_id),
    41
DELETE /storekeeper/api/units/(int:id),
    44
DELETE /storekeeper/api/users/(int:id),
    47
DELETE /storekeeper/api/users/(int:id)/config/(string:name),
    49
DELETE /storekeeper/api/vendors/(int:id),
    52
DELETE /storekeeper/api/works/(int:id),
    56
DELETE /storekeeper/api/works/(int:id)/items/(int:item_id),
    61
```



## A

admin, [10](#)

api, [12](#)

## C

client, [10](#)

## D

dependencies, [9](#)

## F

Flask-Admin, [10](#)

## P

package.sh, [9](#)

## R

RCP API, [12](#)

## S

server, [10](#)

sql, [63](#)